FORESIGHTED PEOPLE FINDING AND FOLLOWING

Dissertation

zur

Erlangung des Doktorgrades (Dr. rer. nat.)

der

Mathematisch-Naturwissenschaftlichen Fakultät

der

Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

ABDELMONIEM BAYOUMI

aus

Giza, Ägypten

Bonn, April 2018

Angefertigt mit Genehmigung der
Mathematisch-Naturwissenschaftlichen Fakultät der
Rheinischen Friedrich-Wilhelms-Universität Bonn

1. GUTACHTER: Prof. Dr. Maren Bennewitz
2. GUTACHTER: Prof. Dr. Jürgen Gall

TAG DER PROMOTION: 02.07.2018
ERSCHEINUNGSJAHR: 2018

Families are the compass that guides us. They are the inspiration to reach great heights, and our comfort when we occasionally falter.

— Brad Henry

Dedicated to my family.

ABSTRACT

Mobile service robots are needed in several applications (e.g., transportation systems, autonomous shopping carts, household activities . . . etc). In such scenarios the robot aids the user with tasks that require the robot to move freely across the environment in addition to direct interaction at certain times. Therefore, such a robot needs a strategy to quickly find the user whenever needed, in addition to a strategy that enables the robot to reason about the user's intended destination to be able to follow him in a foresighted manner if the user needs its help at that destination. In this dissertation, we tackle each of those problems separately in a divide and conquer manner.

We present an approach to learn optimal navigation actions for assistance tasks in which the robot aims at efficiently reaching the final navigation goal of a human where service has to be provided. Always following the human at a close distance might hereby result in inefficient trajectories, since people regularly do not move on the shortest path to their destination (e.g., they may move to grab the phone or make a note). Therefore, a service robot should infer the human's intended navigation goal and compute its own motion based on that prediction. We propose to perform a prediction about the human's future movements and use this information in a reinforcement learning framework to generate foresighted navigation actions for the robot. Since frequent occlusions of the human will occur due to obstacles and the robot's constrained field of view, the estimate about the humans's position and the prediction of the next destination are affected by uncertainty. Our approach deals with such situations by explicitly considering occlusions in the reward function such that the robot automatically considers to execute actions to get the human in its field of view. We show in simulated and real-world experiments that our technique leads to significantly shorter paths compared to an approach in which the robot always tries to closely follow the user and, additionally, can handle occlusions.

On the other side, an autonomous robot that directly helps users with certain tasks often first has to quickly find a user, especially when this person moves around frequently. A search method that relies on a greedy approach that do not perform any predictions about the user's most likely location, even when it is provided with background information about the frequently visited destinations of the user, might not be the best option. In this dissertation, we propose to compute the likelihood of the user's observability at each possible location in the environment based on simulations that rely on hidden Markov model based predictions. As the robot needs time to reach the search locations, we take this time into account as well as the visibility constraints. In this way we aim at selecting effective search locations for the robot to find the user as fast as possible. As our experiments in various simulated environments show, our approach leads to significantly shorter search times compared to the greedy approach.

ZUSAMMENFASSUNG

Mobile Serviceroboter werden in verschiedenen Anwendungen benötigt (z.B. Transportsysteme, autonome Einkaufswagen, Haushaltsaktivitäten usw.). In solchen Szenarien unterstützt der Roboter den Benutzer mit Aufgaben, bei denen sich der Roboter zusätzlich zur direkten Interaktion zu bestimmten Zeiten frei in der Umgebung bewegen muss. Daher benötigt ein solcher Roboter eine Strategie, um den Benutzer bei Bedarf schnell zu finden. Außerdem benötigt der Roboter eine Strategie, die es dem Roboter ermöglicht über das beabsichtigte Ziel des Benutzers nachzudenken, um ihm vorausschauend folgen zu können, wenn der Benutzer seine Hilfe an diesem Ziel benötigt. In dieser Dissertation gehen wir jedes dieser Probleme auf getrennte Weise an.

Wir stellen einen Ansatz vor, um optimale Navigationsaktionen für Assistenzaufgaben zu erlernen, bei denen der Roboter darauf abzielt, das endgültige Navigationsziel eines Menschen, an dem der Dienst erbracht werden muss, effizient zu erreichen. Wenn man dem Menschen in unmittelbarer Nähe folgt, kann dies zu ineffizienten Bewegungsbahnen führen, da sich Menschen nicht regelmäßig auf dem kürzesten Weg zu ihrem Ziel bewegen (z. B. können sie sich bewegen, um das Telefon zu ergreifen oder eine Notiz zu machen). Daher sollte ein Serviceroboter das beabsichtigte Navigationsziel des Menschen ableiten und basierend auf dieser Vorhersage seine eigene Bewegung berechnen. Wir schlagen vor, eine Vorhersage über die zukünftigen Bewegungen des Menschen durchzuführen und diese Information in einem Verstärkungslernsystem zu verwenden, um vorausschauende Navigationsaktionen für den Roboter zu erzeugen. Da aufgrund von Hindernissen und dem eingeschränkten Sichtfeld des Roboters häufige Verdeckungen des Menschen auftreten, sind die Schätzungen über die Position des Menschen und die Vorhersage des nächsten Ziels von Unsicherheit betroffen. Unser Ansatz behandelt solche Situationen, indem er Okklusionen in der Belohnungsfunktion explizit berücksichtigt, so dass der Roboter automatisch Aktionen ausführt, um den Menschen in sein Sichtfeld zu bringen. Wir zeigen in simulierten und realen Experimenten, dass unsere Technik zu wesentlich kürzeren Wegen führt als bei einem

Ansatz, bei dem der Roboter immer versucht, dem Benutzer genau zu folgen, während unserer Ansatz zusätzlich mit Okklusionen umgehen kann.

Auf der anderen Seite muss ein autonomer Roboter, der Benutzern mit bestimmten Aufgaben direkt hilft, oft erst einen Benutzer finden, wenn er dazu aufgefordert wird, besonders wenn sich diese Person häufig bewegt. Eine Suchmethode, die auf einem gierigen Ansatz beruht und keine Vorhersagen über den wahrscheinlichsten Standort des Benutzers trifft, selbst wenn es mit Hintergrundinformationen über die häufig besuchten Ziele des Benutzers bereitgestellt wird, ist möglicherweise nicht die beste Option. In dieser Dissertation schlagen wir vor, die Wahrscheinlichkeit der Beobachtbarkeit des Benutzers an jedem möglichen Ort in der Umgebung basierend auf Simulationen zu berechnen, die auf Hidden-Markov-Modell-basierten Vorhersagen beruhen. Da der Roboter Zeit braucht, um die Suchorte zu erreichen, berücksichtigen wir diese Zeit sowie die Sichtbarkeitseinschränkungen. Auf diese Weise zielen wir darauf ab, effektive Suchorte für den Roboter auszuwählen, um den Benutzer so schnell wie möglich zu finden. Wie unsere Experimente in verschiedenen simulierten Umgebungen zeigen, führt unser Ansatz zu wesentlich kürzeren Suchzeiten im Vergleich zur Lösung des gierigen Ansatzes.

No one knows what the right algorithm is, but it gives us hope that if we can discover some crude approximation of whatever this algorithm is and implement it on a computer, that can help us make a lot of progress.

— Andrew Ng

ACKNOWLEDGEMENTS

First and foremost, I would like to express my gratitude to Prof. Dr. Maren Bennewitz for giving me the opportunity to work in her laboratory. I would like to specially thank her for her generous support, foresighted advice and constructive criticism without which this work would have never been completed. I thank my colleague Philipp Karkowski for the fruitful meetings and discussions. Furthermore, I thank Stefan Oßwald, Peter Regier, Dr. Marcell Missura, Philipp Bruckschen and all the others in the Humanoid Robots Lab for all their help and support. I thank my family for their unconditional support and love which have been backing me along my whole life.

This work has been supported by the German Academic Exchange Service (DAAD) and the Egyptian Ministry for Higher Education as well as by the European Commission under contract number FP7-610532-SQUIRREL and by the DFG Research Unit FOR 2535 Anticipating Human Behavior.

CONTENTS

1 INTRODUCTION 1
- 1.1 Key Contributions . . . 6
- 1.2 Publications . . . 7

2 RELATED WORK 9
- 2.1 People Following . . . 9
- 2.2 People Finding . . . 12

3 INTRODUCTION TO REINFORCEMENT LEARNING 15
- 3.1 Introduction . . . 15
- 3.2 Elements of Reinforcement Learning . . . 17
  - 3.2.1 Policy . . . 17
  - 3.2.2 Reward Function . . . 17
  - 3.2.3 Value Function . . . 18
  - 3.2.4 Model . . . 19
- 3.3 Optimal Value Functions . . . 20
- 3.4 SARSA(λ) . . . 20
- 3.5 Comparisons & Discussions . . . 24
  - 3.5.1 Dynamic Programming Techniques . . . 24
  - 3.5.2 Monte Carlo Techniques . . . 24
  - 3.5.3 Temporal-Difference (TD) Learning . . . 25
  - 3.5.4 Eligibility Traces . . . 25
- 3.6 Conclusion . . . 27

4 FORESIGHTED PEOPLE FOLLOWING WITH KNOWN POSES 29
- 4.1 Introduction . . . 29
  - 4.1.1 Contribution . . . 30
  - 4.1.2 Assumptions . . . 31
  - 4.1.3 Framework Overview . . . 32
  - 4.1.4 Organization . . . 34
- 4.2 Motion Prediction . . . 34
- 4.3 Learning Navigation Actions . . . 36
  - 4.3.1 State Space $\mathcal{S}$ . . . 37
  - 4.3.2 Action Set $\mathcal{A}$ and Q-Table . . . 37
  - 4.3.3 Reward $\mathcal{R}$ . . . 38
  - 4.3.4 Reinforcement Learning . . . 39
- 4.4 Experimental Results . . . 40
  - 4.4.1 Environment Setup . . . 40
  - 4.4.2 Trajectory Generation for Training and Testing . . . 41

4.4.3 Parameters 42
4.4.4 Evaluation Metrics 43
4.4.5 Experiments in Simulation 43
4.4.6 Experiments with a Real Robot 46
4.5 Conclusions 47
5 FORESIGHTED PEOPLE FOLLOWING UNDER OCCLUSIONS 49
5.1 Introduction 49
5.1.1 Contributions 50
5.1.2 Framework Overview 51
5.1.3 Organization 53
5.2 Problem Formulation 54
5.3 Motion Prediction 54
5.4 Learning Navigation Actions 56
5.4.1 State Space $\mathcal{S}$ 56
5.4.2 Action Set $\mathcal{A}$ and Q-tables 58
5.4.3 Reward Function $\mathcal{R}$ 58
5.4.4 Reinforcement Learning 59
5.5 Experiments 61
5.5.1 Experimental Setup 61
5.5.2 Experiments in Simulation 62
5.5.3 Real-World Experiment 64
5.6 Conclusions 66
6 PEOPLE FINDING UNDER VISIBILITY CONSTRAINTS 69
6.1 Introduction 69
6.1.1 Contributions 70
6.1.2 Framework Overview 70
6.1.3 Organization 72
6.2 Problem Formulation 73
6.3 Graph-Based People Tracking 73
6.4 Selecting Search Locations 76
6.4.1 Using HMM-based Simulations 77
6.4.2 Extended Version of HMM-based Simulations 79
6.5 Experimental Results 82
6.5.1 Experimental Setup 82
6.5.2 Results and Evaluation 83
6.6 Conclusions 86
7 CONCLUSIONS & OUTLOOK 89

BIBLIOGRAPHY 93

LIST OF FIGURES

Figure 1 Future conception of cognitive robots operating in hazardous environments 2
Figure 2 A conception of a service robot operating in domestic domain 2
Figure 3 Children with autism can engage better with robots more than humans 3
Figure 4 ASIMO pushes a cart to deliver some items in an office environment 4
Figure 5 The interaction during the reinforcement learning process between the agent and the environment 16
Figure 6 An overview of our learning framework. . 33
Figure 7 Maps used for the experiments 42
Figure 8 Experiment in which the user first makes a large detour before walking toward their intended destination 45
Figure 9 The user performed an inefficient trajectory by walking around the obstacle in the left part instead of directly going to his goal location 46
Figure 10 Trajectories of the human and the robot corresponding to the real-world experiment recorded by the MoCap system . . . 47
Figure 11 An overview of our proposed system . . . 51
Figure 12 Example of foresighted people following 53
Figure 13 Simulation environment with topo-metric graph. The environment is represented as a grid map with an overlaid topo-metric graph 57
Figure 14 The real-world environment with an overlaid topo-metric graph 61
Figure 15 Common human trajectories used in the simulation environment 63
Figure 16 The robot and the user with a board of ArUco markers to simplify pose estimation in the real-world experiment. 65
Figure 17 Real-world experiment 65

Figure 18 Example of selecting a search location with maximum observability 71
Figure 19 Three simulation environments with overlaid topo-metric graphs 74
Figure 20 An example showing a selected search location according to our HMM-based simulations 79
Figure 21 Average relative search time achieved by both versions of our HMM-based simulations approach with respect to a greedy approach with background information . 85
Figure 22 Average relative search time achieved by both versions of our HMM-based simulations approach with respect to the one-step to estimation method 85

LIST OF TABLES

Table 1 Gain in traveled distance and completion time 44
Table 2 Path length reduction of our approach compared to directly following 63
Table 3 Path length reduction of a prediction heuristic compared to directly following 64
Table 4 Percentage of switching to the greedy approach. 86

LIST OF ALGORITHMS

1 SARSA(λ) 23

2 Selection of the next search location using HMM-based simulations 78

1 INTRODUCTION

Mobile robots have gained more attention in the last decade in conjunction with the significant advancement in the artificial intelligence and the computer vision fields. The cognitive capabilities of mobile robots were significantly enhanced which accordingly widens the scope of their applications. Mobile robots can be deployed, using their cognitive capabilities, as service robots to help humans in difficult or dangerous tasks, i.e., transporting heavy items, disposal of dangerous wastes, or even working in normal landfill sites, as shown in Figure 1. Additionally, such robots are needed in domestic domains to offer care for elderly people and patients in addition to offering help for users in daily tasks (e.g., shopping, cleaning ... etc), as illustrated in Figure 2. Furthermore, robots that can interact and play with children have shown a remarkable success in helping children with autism more than humans. Since robots are simple and predictable, the children can engage better with them and enhance their social interaction skills [Kim et al., 2013], as illustrated in Figure 3.

Smart service robots are crucially needed to help humanity.

Mobile service robots that can quickly find people and efficiently follow them are needed in several applications such as in industrial settings and in work environments where robots can be deployed as transportation systems for heavy items (see Figure 4). Also, they can be deployed in home scenarios, especially for assisting the elderly people in daily housekeeping activities. Moreover, robots with such features may act as autonomous shopping carts in stores. In such scenarios the robot aids the user with tasks that require the robot to move freely across the environment in addition to direct interaction at certain times. Therefore, such a robot needs a strategy to find the user as fast as possible when it is necessary, in addition to a strategy that enables the robot to reason about the user's intended destination in order to follow them in a foresighted manner. Resolving occlusions that may occur during the following task, due to obstacles in the environment, is a key challenge that the robot must overcome via inferring the user's intended destinations, as well as executing navigation actions that ex-

People finding and following are key features for service robots.

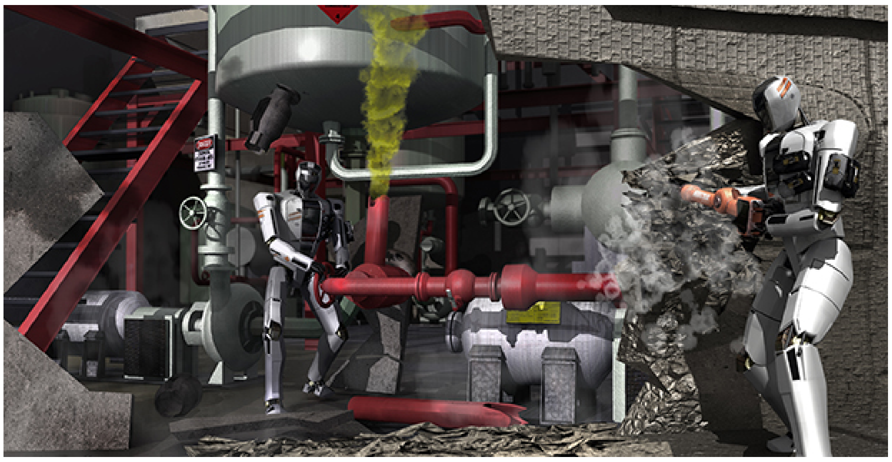


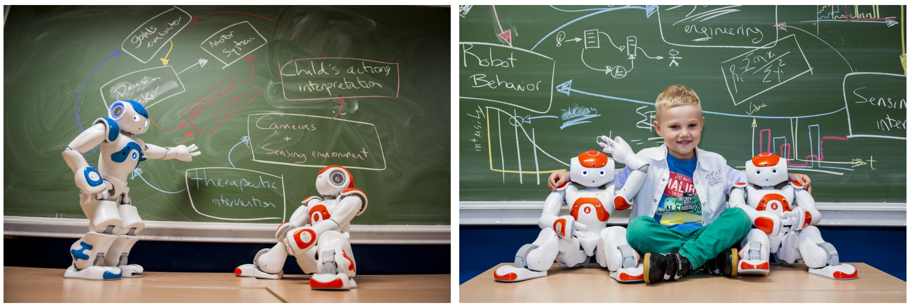
Figure 1: Future conception of cognitive robots operating in hazardous environments that are not suitable for humans (Source: DARPA).

Figure 2: A conception of a service robot (Amigo) operating in domestic domain (Source: Tech United Eindhoven): (a) to take care of patients in hospitals, and (b) to help users in shopping at supermarkets.

ploits the environment structure to gain information that confirms or refutes the belief about the user's intended destination, especially when the user is out of the robot's field of view. Additionally, the robot must be able to resolve the situations in which the user takes paths that are impassable for the robot due to size or safety constraints via finding alternative paths while maintaining an updated belief about the user's intended destination.

Maximum coverage techniques lead to unnecessary delays.

Unfortunately, the existing people finding and following approaches still suffer from unnecessary delays and inefficiency, respectively. Concerning people finding techniques that try to maximally cover the visible area of the environment [Choset, 2001; Guibas et al., 1996; Suzuki and Yamashita, 1992], they



(a) (b)

Figure 3: (a) Nao robots can be deployed to play simple games with children. (b) Children with autism can engage better with robots more than humans which accordingly improves their social interaction skills (Source: University of Portsmouth and Science X).

lead to long search times and high navigation costs as they aim at covering the whole environment without estimating the user's location. Moreover, the maximum coverage techniques will not necessarily revisit already covered regions and since the user is assumed to move freely across the environment, the robot might miss them during the search. Therefore, it will be much more efficient to exploit the environment 's visibility characteristics as well as making use of prior knowledge about the user's potential destinations which will lead to finding the user faster.

Furthermore, following the user at a certain distance [Harmati and Skrzypczyk, 2009; J. Huang et al., 2006; L. Huang, 2009; Kuderer and Burgard, 2014; Nascimento et al., 2013; Pradhan et al., 2013] ignoring that humans may not always take the shortest path to their next destination might lead to an inefficient robot navigation behavior in scenarios where the robot is needed to encounter the user only at designated places. In such situations, robots that directly follow the user without any reasoning about their intended destination will suffer from unnecessary battery consumption as well as faster actuators' wear. Additionally, the user might occasionally move through passages that are impassable for the robot such that the robot is forced to find an alternative route. In such cases the problem arises that the user will eventually be out of the robot's field of view, which will lead to an uncertain estimate about the user's position and a wrongly predicted destination. Thus, the robot will need to consider active re-localization of the user,

Direct following may lead to inefficient trajectories.

Figure 4: ASIMO pushes a cart to deliver some items in an office environment (Source: Honda).

via exploiting knowledge about the user's common paths in the environment, to improve the estimate and to be able to infer the next navigation goal.

In order to address the aforementioned literature gaps, a successful people finding and following strategy must be able to autonomously infer foresighted answers for the following questions without the need for any human interference:

Foresighted strategies will enhance the efficiency of the people finding and following tasks.

- How can the robot estimate the user's potential locations at any time step when the user is out of the robot's field of view? How to evaluate the likelihood of each potential location?
- Does the robot have enough time to reach that estimated location, i.e., to search for the user there, from its current location, before the user moves again?
- In which cases does the robot have to closely follow the user if the user is within the robot's field of view? When is it better to take an alternative path (or just keep waiting at its current location) to minimize navigation costs even if this will lead to losing track of the user?
- How to predict the user's intended final destination?
- How to select navigation actions that can improve the robot's belief about the user's predicted destination even if the user is out of the robot's field of view?

To answer these aforementioned questions, we need to exploit the prior knowledge about the environment structure, as well as the user's set of common paths. Accordingly, we can simulate various future scenarios of the user's behavior and estimate the user's most likely current location as well as their intended destination. These estimates are then updated based on the robot's observations which depend on our selected navigation actions. Therefore, we have to consider selecting key navigation actions which will lead to more information gain as well.

This dissertation is organized as follows. We first introduce a literature review in Chapter 2 about existing approaches for both people following and people finding, respectively. Furthermore, we point out the gaps in these approaches and how our novel approaches address them. Then, in Chapter 3 we introduce the main concepts of reinforcement learning and demonstrate the properties of the various reinforcement learning techniques. Furthermore, we hold a comparison between these demonstrated techniques and justify our choice of using SARSA(λ) in the next two chapters.

After that, we discuss in Chapter 4 our proposed learning framework for foresighted people following given known poses of both the robot and the user. This assumption about known poses can be achieved in reality via an external motion capture systems that covers the environment within which the following task takes place. In this framework, we focus on exploring the feasibility of applying a learning approach on the following problem and whether it is capable of generating navigation actions that leads to foresighted following behavior. This foresighted following behavior implies that the robot infers about the user's intended destination and does not follow the user to locations where they do not need the robot's help. The robot tends to meet the user at their intended final destination through a more efficient path (that fits the robot size and satisfies any further constraints) instead of the direct naive following of the user, which leads accordingly to a decreased traveled distance by the robot in such scenarios.

People following given known poses.

In Chapter 5, we extend our people following framework in order to attain a performance that better fits real-world scenarios without involving any assumptions about known poses of the user. In other words, the robot has to rely on its sensors to localize itself as well as tracking the user. Therefore, the robot is

People following under occlusions.

vulnerable to occlusions due to its limited field of view as well as obstacles that restrict this limited field of view even more. Additionally, occlusions may occur if the user walks through paths which are impassable for the robot due to size or safety constraints, where in such scenarios a robot that performs direct following will get stuck at the entrance of such locations. Therefore, the robot applying our approach will have to find an alternative path to resume its task, which accordingly will lead to losing track of the user and thus much more uncertainty about the estimated destination will be involved. Therefore, we enhanced our aforementioned framework approach in order to improve the generated foresighted navigation behavior to reason better about such situations.

After that, we introduce our proposed people finding approach in Chapter 6. It focuses on scenarios where the user moves along common paths between places where they remain for a while, e.g., to discuss work with a colleague, grab material from the printer, or get a coffee. The service robot aids a user at tasks that require direct interaction. However, the robot may also move freely across the environment in order to accomplish its tasks. Furthermore, the user's initial location is unknown to the robot and the robot may suffer from noisy sensors and dynamic obstacles that occlude its limited field of view. Our approach addresses the deficiencies arising from the existing approaches which focus either on maximal coverage of the environment while only considering the visibility characteristics of the environment without making use of any prior knowledge, or they predict the user's location and navigate to that predicted location. Accordingly, we make use of prior knowledge about possible destinations of the user and their connecting paths to achieve a short searching time. Our approach simulates possible behaviors of the user for future time steps in addition to considering the visibility characteristics of the environment in order to select a good search location at which there is a high likelihood to observe the user.

People finding.

1.1 KEY CONTRIBUTIONS

In this dissertation we discuss several contributions in the field of navigation with respect to the people following as well as the people finding tasks. In this section we summarize the key contributions as follows:

- to the best of our knowledge, we present the first solution for applications involving people following tasks that considers the efficiency of the generated robot paths (Chapter 4 & Chapter 5),
- the generation of foresighted robot behavior for people following based on learned navigation strategies and the prediction of human motion (Chapter 4 & Chapter 5),
- a foresighted people following framework that relies only on the robot's on-board sensors and can deal with occlusions as well as impassable passages for the robot (Chapter 5),
- a learning framework that is able to handle also large environments without overloading the learning process of foresighted navigation actions for people following tasks (Chapter 5),
- a novel people finding approach that computes the likelihood of the user's observability at each possible location based on HMM-based simulations (Chapter 6),
- achieving a short search time via making use of prior knowledge about frequently visited destinations of the user and their typical paths (Chapter 6),
- determining good search locations using a hidden Markov model on a graph representation of the moving possibilities in the environment (Chapter 6),
- taking into account the time needed by the robot to reach the search locations from its current position as well as the visibility constrains that arise from obstacles (Chapter 6).

1.2 PUBLICATIONS

Parts of this dissertation have been published before [Bayoumi and Bennewitz, 2015; 2016; Bayoumi, Karkowski, and Bennewitz, 2017; 2018; submitted in 2018]. We present the publications in chronological order as follows:

- A. Bayoumi, P. Karkowski, and M. Bennewitz. People Finding under Visibility Constraints using Hidden Markov Models. In *Robotics & Autonomous Systems*, submitted in 2018
- A. Bayoumi, P. Karkowski, and M. Bennewitz. People Finding under Visibility Constraints using Graph-Based Mo-

tion Prediction. In *Proc. of the Int. Conf. on Intelligent Autonomous Systems (IAS)*, 2018, accepted

- A. Bayoumi, P. Karkowski, and M. Bennewitz. Learning Foresighted People Following under Occlusions. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2017
- A. Bayoumi and M. Bennewitz. Learning optimal navigation actions for foresighted robot behavior during assistance tasks. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2016
- A. Bayoumi and M. Bennewitz. Efficient Human Following Using Reinforcement Learning. In *Proc. of the IROS Workshop on Machine Learning in Planning and Control of Robot Motion (MLCP)*, 2015

2 RELATED WORK

In this chapter we review the related work to the people following and finding tasks, respectively. We review the existing approaches of each task independently and discuss the literature gaps within them. Furthermore, we highlight how our proposed novel approaches address such gaps.

2.1 PEOPLE FOLLOWING

Approaches based on control theory.

The task of people following and tracking has been thoroughly investigated using control theory, e.g., J. Huang et al. [2006] introduced an approach using velocity control that guides the robot along the trajectory of a Bézier curve to a leading robot. Furthermore, L. Huang [2009] presented two control models for both the linear and angular velocity of a robot. These control models focus on originating velocity commands that allow the robot to follow a human while ensuring smoothness of the resulting trajectory. Harmati and Skrzypczyk [2009] applied a fuzzy logic controller and proposed a game-theory based method for a team of robots tracking a target. In this approach, each robot has knowledge about the positions of its teammates and optimizes its control signals using a cost function that takes also into account the predicted decisions of the other robots. Similarly, Nascimento et al. [2013] developed an approach for collaborating robots' formation control during tracking of a moving target or a leader. They use a nonlinear predictive formation control model in a distributed architecture for collaborating robots. Each robot shares information about its pose with the rest of the team and optimizes its control signals under a prediction of the next states of the target as well as of the other team members. Pradhan et al. [2013] proposed a path planning method with a navigation function that uses predictive fields of moving obstacles to follow a target. Choi et al. [2010] presented a model for the following task in an environment equipped with RFID tags that are used by the robots for localization. The velocity of the following robot is computed according to the leading robot's relative position, it is inversely proportional to the relative distance.

Fixed distance following.

Furthermore, there are approaches that aim at following a human with a fixed distance, e.g., Nishimura et al. [2007] developed a modified shopping cart to follow a person in a certain range. Prassler et al. [2002] considered side-by-side following within the application of a robotic wheelchair following a human. Morales et al. [2012] analyzed side-by-side trajectories of humans in order to predict the trajectory of a human during walking beside a robot. Using the learned utility function, the optimal robot trajectory can then be planned locally.

Machine learning techniques.

Regarding the use of machine learning techniques in related problems, Goldhoorn et al. [2014] introduced a two-stage approach to find a moving person first and then following them. The authors presented continuous real-time partially observable Monte Carlo planning (CR-POMCP) to generate a policy tree through simulations. Afterwards, the authors extended the CR-POMCP by combining it with a standard direct following approach. Thus, the robot moves directly to the human as long as it is in the field of view. If the human is not visible, the authors use the human position with the highest probability according to the CR-POMCP as the next navigation goal of the robot. In contrast, our approach reasons about the next destination of the human and uses the learned policy to decide on the best action, thereby explicitly taking into account observation actions. Our system does the reasoning in both cases whether the human is in the field of view or not to generate efficient robot behavior.

Kretzschmar et al. [2014] learned a probabilistic model of pedestrians from observing their trajectories. Kuderer and Burgard [2014] applied that approach to predict a human's trajectory and plan an intelligent path for a robot following the human. The proposed method takes into account information about obstacles in the environment and computes the robot's trajectory so that the distance to the desired relative position along the predicted human path is minimized. In our work, we go beyond such an idea and take into account long-term prediction of the human's motions and compute efficient robot actions, instead of staying always in the vicinity of the human.

Contribution.

The contribution of our work is the generation of foresighted robot behavior based on learned navigation strategies and the prediction of human motion. The prediction of human motions itself is not the focus. Nevertheless, we present some recent work in this area in the following. Best and Fitch [2015] presented a Bayesian trajectory prediction model for moving people. They assume that there is a set of predefined destinations between which the person moves on the shortest path and, therefore, the likelihood of a movement is inversely proportional to how far is the corresponding future position from the shortest path to a given destination. Ziebart et al. [2009] introduced an approach based on a softened Markov decision process (MDP) that is trained on a set of observed human trajectories and uses a Q-table to encode the human's movements in a grid map at any point at any time. The authors proposed to make use of this modeling for planning collision-free paths. Vasquez [2016] extended the work of Ziebart et al. [2009] in order to decrease the computational complexity. Both, Vasquez and Ziebart et al. assume that the cost function based on which the human plans its motion is given a priori. In Chapter 4, we implemented the technique of Ziebart et al. [2009] but noticed in our experiments that we could not achieve good prediction results in case the human deviates from learned trajectories. Therefore, we apply a particle filter based motion prediction in Chapter 5.

Motion prediction.

Further approaches predict human trajectories to generate robot motions that do not interfere with people. Bennewitz et al. [2005] proposed to predict human trajectories based on learned motion patterns to avoid interferences in tight environments. Laugier et al. [2007] use hidden Markov models to represent the motion of dynamic obstacles and avoid collision. Fentanes et al. [2015] proposed to predict periodic changes in the environment that affect the navigation actions of the robot. They compute a frequency map and combine it with the topological map of the environment.

Conclusion.

In contrast to all the approaches discussed above, we present a learning framework in Chapter 4 that enables a service robot to efficiently reach the intended destination of the human. Using our approach, the robot is not required to follow the human with a fixed distance, which might lead to inefficient trajectories, instead the robot predicts the navigation goal, constantly updates it, and adapts its actions based on a learned strategy via reinforcement learning that is robust to prediction errors. Moreover, the extension of our proposed framework (Chapter 5) relies only on the robot's on-board sensors and can deal with occlusions. Moreover, with the modified state space in that extension we are able to handle also large environments without overloading the learning process.

2.2 PEOPLE FINDING

Maximum coverage.

The problem of finding a moving person in an environment was early studied as a coverage problem based on the robot's visibility polygon, e.g., Suzuki and Yamashita [1992] and Guibas et al. [1996] consider finding intruders by a surveillance robot. They make use of the geometrical properties of the robot's visibility polygon in order to find a directed graph on which a pursuer take a tour to clear the environment from unpredictable intruders. They use the geometric properties of the visibility polygon in order to minimize the visited points through the environment. However, they do not predict motion of the person and thus they do not estimate the person's pose. Furthermore, a survey about different coverage approaches is presented by Choset [2001].

Multi-robot cooperation.

Concerning making use of multi-robot cooperation for the purpose of a coverage problem, Moors et al. [2005] propose a graph-based approach for multi-robot coordination in order to clear a certain indoor area from some undetected intruders. They represent the environment using a topometric representation. Then, they divide this graph into a group of sub-graphs where the size of each sub-graph should be less than a certain threshold that depends on the number of available robots. Some of the robots are used as guards in the connections between these sub-graphs, i.e., doors and corridors, in order to avoid recontamination of cleared areas. Moreover, Kolling and Carpin [2008] present a multi-robot strategy in order to solve the graph-clearance problem in NP-complete way. However, they do not consider the encountering probability of intruders, i.e., they have to check every single position in the environment. Hollinger, Singh, et al. [2010] assume that they have some knowledge about the target's motion model aiming at achieving a balance between getting a guaranteed solution and efficiency. Therefore, they allocate the searching robots for the clearance schedule such that some of the searching robots consider a worst case scenario in which they have no knowledge about the user's motion model, while the others make use of the given information. Then, Hollinger, Yerramalli, et al. [2015] consider data fusion between the searching robots under uncertain communication in order to share their estimates about their target's location.

Furthermore, Stiffler et al. [2017] additionally consider the problem of unreliable sensors. The authors developed a visibility-based geometric formulation to place the surveillance robot at specific environment locations that maximize the traveled distance by the intruder through the robot's visible region to increase the likelihood of observing the intruder by the unreliable sensors. All these solutions to the coverage problem do not predict motion of the person and thus cannot provide any pose estimate. They lead to long search times and high navigation costs as they aim at covering the whole environment.

Motion prediction.

On the other hand, several approaches that predict motions and aim at minimizing the searching time for a mobile robot have been presented. Tipaldi and Arras [2011] proposed to learn a spatial affordance map in order to increase the probability for the robot to encounter specific humans. They apply a Poisson process to relate space, time, and occurrence probability of activity events. Afterwards, the spatio-temporal model can be used to generate an optimal path on a grid map of the environment for a mobile robot to encounter specific humans. This approach does not make use of any sensor modalities to update the belief about the location of a user but considers just the encounter probability of grid cells. Schwenk et al. [2014] developed a search approach that uses a highly abstract topological representation of the environment and learns about the user's behaviors in order to estimate the likelihood of the user's current room. Here, it is assumed that the robot detects people when they are within a range of 1.8 m around the robot. Kulich et al. [2016] introduced a model that learns the temporal likelihood of possible desired interactions to actively search for humans in order to interact with them in public space. Krajník et al. [2015] presented a method based on spatio-temporal models to enable the robot finding non-stationary objects in an office environment. The authors represent the environment as an abstract topological map and combine it with periodic functions in order to compute the likelihood of existence of the objects at any node of the map with respect to the time. All these approaches, however, ignore the visibility constraints resulting from the environment layout.

Other approaches considered the frequency of human existence at specific locations as well as the robot's field of view.

Existence frequency.

The idea here is to construct a probability distribution for every hour of the day. For example, in the work of Volkhardt and Gross [2013], the robot searches for the human at pre-defined locations, where each location is assigned a probability relative to the frequency of observing the human there. Accordingly, the robot selects the location with the highest probability. Mehdi and Berns [2014] presented a technique that generates a minimum set of view points that ensure a maximum coverage of the environment. The authors proposed to construct a probability distribution about the human's observability at these destinations during each hour of the day and take the navigation cost into account while deciding which one of the view points to take as search location. These approaches do not model the human's motion and therefore cannot predict their expected position at a certain intermediate time step.

Goldhoorn et al. [2017] proposed using particle filters to estimate the most likely location of the user at the current time step. The robot moves toward that location for few time steps then updates its estimate about the user's position and recomputes the robot's movement. As opposed to our method, this technique does not take into account the time needed by the robot to reach search locations from its current place. Moreover, moving the robot just for few time steps and then selecting another search location often leads to oscillating navigation behavior as the estimation jumps across the map as we realized in our experiments.

Conclusion.

In contrast to all the mentioned search methods, our system in Chapter 6 models the human's motion and provides a probability distribution about their position at each time step. We consider the robot's limited field of view and visibility constraints when computing the likelihood of observing the user at a certain place and also take into account the time needed by the robot to reach the search locations.

3 INTRODUCTION TO REINFORCEMENT LEARNING

In this chapter, we briefly introduce the basic concepts of reinforcement learning techniques. After that, we present the SARSA(λ) approach that we use in our proposed approaches in Chapter 4 and Chapter 5. Furthermore, we compare it to various reinforcement learning techniques showing the advantages and disadvantages of each of them and, thus, we justify our choice of the SARSA(λ) approach to be used by our proposed approaches. More detailed explanations can be found in the work of Kaelbling et al. [1996] and Sutton and Barto [1998].

3.1 INTRODUCTION

Reinforcement learning is a type of learning opposed to two other main types of learning: supervised and unsupervised learning, where supervised learning depends on labeled examples, i.e., these examples consist of a set of inputs and their corresponding outputs according to some unknown function that we are interested to learn. On the other hand, unsupervised learning tends to learn a generative model of the input data that tries to describe possible patterns and features of these inputs without the need for labeled outputs [Ghahramani, 2004]. Reinforcement learning is a learning process through interaction with a certain learning environment (as illustrated in Figure 5), where a set of possible actions are given from which the learning agent can choose one to be executed at each time step. These actions can change the state of the environment when they are executed. As opposed to supervised learning, the perfect actions for each situation are not given. Instead, the agent has to learn from its previous mistakes and construct some form of experience based on which it can select better actions. The learning agent is given some evaluation measurements instead to help in evaluating the interaction with the environment during the learning process, i.e., the execution of the actions change the state of the learning environment and based on the evaluation of that change a reward or a penalty is

Types of learning.

given to the learning agent. Thus, reinforcement learning tends to maximize the achieved rewards which leads to learning the best actions that fit every state of the environment.

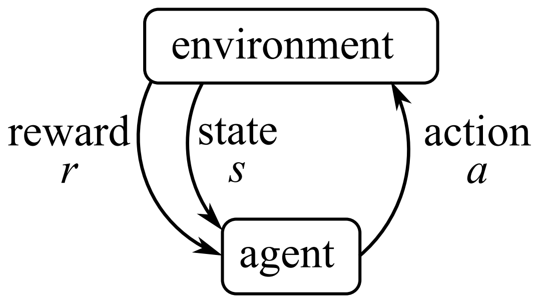


Figure 5: The interaction during the reinforcement learning process between the agent and the environment.

MDP.

Furthermore, a reinforcement learning task is defined as a Markov decision process (MDP) if it satisfies the Markov property in which the transition to the next state from its current state as well as the obtained immediate reward from this transition are dependent only on the current state and selected action to be executed at that time step, as follows:

$$\begin{aligned} &\Pr\left\{s_{t+1}=s', r_{t+1}=r \middle| s_t, a_t, r_t, s_{t-1}, a_{t-1}, \ldots, r_1, s_0, a_0\right\} \\ &= \Pr\left\{s_{t+1}=s', r_{t+1}=r \middle| s_t, a_t\right\} \end{aligned} \tag{1}$$

where r_{t+1} is the obtained immediate reward from executing a certain action a_t while being in state s_t at time t to end up in a successor state s_{t+1}.

Exploration vs exploitation.

Additionally, the exploration-exploitation dilemma is a key challenge that faces reinforcement learning techniques. The learning process requires trying various actions for various states in order to learn from that experience, i.e. exploration of the state-action space of the learning problem. However, the process also needs to make use of the so far learned experience in order to enhance it and to maximize the obtained reward, which is unfortunately contradictory. Thus, a good balance between exploration and exploitation has to be found because relying only on either exploration or exploitation alone will lead to the failure of the learning process.

Applications.

Reinforcement learning has been used to solve various problems successfully, e.g., training industrial robots to do certain tasks [Kaelbling et al., 1996], inventory management of supply chains [Giannoccaro and Pontrandolfo, 2002] and dynamic pricing of products based on demand and supply [Busoniu et al., 2008]. Furthermore, solving the Go game by the AlphaGo agent is one of the most famous problems to which reinforcement learning contributed via a training stage of the neural network, that generates the game decisions, on a dataset generated from self-play of reinforcement learning [Silver et al., 2016].

3.2 ELEMENTS OF REINFORCEMENT LEARNING

In this section we present the four main components from which any reinforcement learning process is composed, as follows.

3.2.1 *Policy*

The policy of reinforcement learning agent determines how this agent behaves in different situations, i.e., it defines how an action is selected to be executed in a specific state. The policy can range from a simple lookup table to a very computationally expensive function, where its sole goal is to map each possible state of the environment to its corresponding action. Therefore, the goal of reinforcement learning is to learn a good policy.

Actions' selection probabilities.

A policy is denoted as $\pi(s, a)$ which is the probability of choosing an action a to be executed if the learning environment is experiencing a state s, where s is a state out of the set of possible states $\mathcal{S}$ and a is an action out of the set of possible actions in such state $\mathcal{A}(s)$.

3.2.2 *Reward Function*

The reward function defines the goal of the reinforcement learning process. It maps every state-action pair to a single scalar value, i.e., the immediate reward, that evaluates such pair. In other words, the reward function contributes to the learning process via evaluating the short-term effect of executing a certain action while being in a certain state. The goal of the learning process is to maximize the expected return of these immediate rewards in the long run.

Learning objective.

3.2.3 *Value Function*

The value function defines the value of starting from a certain state of the learning environment in the long run. Thus, the value of a certain state is the expected return to be achieved starting from this state and is denoted as follows:

$$V^{\pi}(s) = E_{\pi}\{R_t | s_t = s\} = E_{\pi}\left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s \right\}, \tag{2}$$

where $V^{\pi}(s)$ is the value of a state s if the agent follows a policy π and E_{π} is the expected value following that policy. Moreover, R_t is the return of rewards at time t and it is defined as follows:

Expected return.

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \tag{3}$$

where r_{t+1} is the immediate reward that is obtained when a certain action is executed at a certain state at time step t. The discounting factor γ is used to favor the impact of immediate rewards of the recent time steps, where its value ranges from 0 to 1. Accordingly, the foresightedness of the learning agent increases as γ approaches 1.

Therefore, a value function is more foresighted compared to the reward function, since a state may yield a low immediate reward, however, it may lead to other future states through which much higher rewards are expected, i.e., delayed reward.

The value function can also represent the value of a state-action pair as well as the value of a state. Accordingly, the value function of a state s and action a following a policy π is denoted as $Q^{\pi}(s, a)$ and is defined as follows:

$$\begin{aligned} Q^{\pi}(s,a) &= E_{\pi}\{R_t | s_t = s, a_t = a\} \\ &= E_{\pi}\left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s, a_t = a \right\}. \end{aligned} \tag{4}$$

Moreover, if the Markov property is satisfied, the value functions can be solved recursively using dynamic programming which is a very effective feature in facilitating the computations.

Thus, the value of any state can be computed using the values of possible successor states, as follows:

$$\begin{aligned} V^{\pi}(s) &= E_{\pi}\{R_t | s_t = s\} \\ &= E_{\pi}\left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s \right\} \\ &= E_{\pi}\left\{ r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \middle| s_t = s \right\} \\ &= \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma E_{\pi}\left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \middle| s_t = s \right\} \right] \\ &= \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V^{\pi}(s') \right], \end{aligned} \tag{5}$$

where $\mathcal{R}^a_{ss'}$ is the expected reward of the state change from s to s' due to executing action a and $\mathcal{P}^a_{ss'}$ is the probability of such transition. Similarly, the value of a state-action pair can be recursively computed as follows:

$$\begin{aligned} Q^{\pi}(s,a) &= E_{\pi}\{r_{t+1} + \gamma V^{\pi}(s_{t+1}) | s_t = s, a_t = a\} \\ &= \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V^{\pi}(s') \right]. \end{aligned} \tag{6}$$

3.2.4 *Model*

The environment model defines the transitions between states according to the executed actions, i.e., $\mathcal{P}^a_{ss'}$ in Equation 6. Therefore, the environment model can be used in planning of tasks, i.e., predicting the next possible states as well as the next rewards. However, in various situations, it might be difficult to have a complete model about the environment, especially in real world tasks. Therefore, the reinforcement learning approaches that do not require any model of the environment and, instead, can learn via direct interaction with the environment, have a big advantage over the methods that need such a model. Furthermore, we discuss later in Section 3.5 which reinforcement learning approaches require having a model and which of them do not.

States' transitions.

3.3 OPTIMAL VALUE FUNCTIONS

In order to solve the reinforcement learning task successfully, we need to find an optimal policy π^* to follow. Such optimal policy is either better or equal to any other policy with respect to the value function of all states as well as for all the state-action pairs, as follows:

$$V^*(s) = \max_{\pi} V^{\pi}(s) \qquad \forall s \in \mathcal{S} \tag{7}$$

Maximizing returns.

and

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a) \qquad \forall s \in \mathcal{S} \quad \forall a \in \mathcal{A}(s). \tag{8}$$

Thus, assuming that our reinforcement learning problem satisfies the Markov property, we can make use of dynamic programming to define the recursive formulas of both optimal value functions, as opposed to the recursive formulas in Equation 5 and Equation 6, as follows:

$$V^*(s) = \max_{a \in \mathcal{A}(s)} \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V^*(s') \right] \tag{9}$$

and

$$Q^*(s,a) = \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma \max_{a'} Q^*(s',a') \right]. \tag{10}$$

In the next section, we will discuss SARSA(λ) approach which is a well-known reinforcement learning method that we use, in the next chapters of this dissertation, to estimate the optimal policy and the optimal function values.

3.4 SARSA(λ)

SARSA(λ) is well known and widely used reinforcement learning technique that does not require having a model of the environment that describes the transition probabilities among all the states. In other words, the learning task takes place via learning from interaction with the environment through real world experiments or simulated ones. These experiments are represented as some learning episodes, where the learning task advances via making use of the returns of the state-action pairs that belong to these episodes. Therefore, SARSA(λ) approach relies on the samples of the state-action pairs that appear during each learning episode and learn from them instead of relying on a model, where a learning episode terminates if a terminal state is reached or after a certain number of elapsed learning steps per episode.

SARSA(λ) relies on temporal-difference (TD) learning. Moreover, it combines TD learning with eligibility traces in order to generalize the learning performance and to increase its efficiency. The TD learning technique updates value functions of the state-action pairs, that appear in the learning episodes, at each step. The key idea behind this technique is to wait for the next time step until the reward for that transition is obtained and thus use the expected return from this transition, i.e., the achieved immediate reward in addition to the current estimated value of the new state, in updating the estimated value of the state that originated that transition. This can be illustrated as follows:

TD Learning.

$$\begin{aligned} V(s_t) &= V(s_t) + \alpha\Big[R_t - V(s_t)\Big] \\ &= V(s_t) + \alpha\Big[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\Big], \end{aligned} \tag{11}$$

where α is a constant step size and $r_{t+1} + \gamma V(s_{t+1})$ is the TD update. Thus, the learning process takes place faster compared to other reinforcement learning techniques that postpone updating the estimated values till the end of each learning episode, e.g., Monte Carlo techniques. However, TD learning does not consider the long run effect of each selected action and its impact on the future return of the learning episode. Therefore, SARSA(λ) combines TD learning with eligibility traces in order to maintain an acceptable speed of learning while generalizing the learning process.

The eligibility traces, in general, are temporary memory variables that record the visits to states over time to determine the responsibility of such visits for future rewards and thus the eligible states are rewarded for their effects, i.e., either positively or negatively. As mentioned above, the eligibility traces tend to balance between the speed of the learning process and its generalization, since focusing only on one of them can be considered as an extreme. Therefore, the eligibility traces introduce an intermediate balanced solution that relies on performing n-step TD predictions, i.e., to wait for n-steps before the update takes place and perform this update based on the immediate rewards obtained from these n-steps instead of waiting until the end of the learning episode. This approach tends to generalize

Balanced approach.

the TD technique and enhance its efficiency as well as its estimates about value functions. Moreover, it will not lead to such slow learning process as in the case of Monte Carlo techniques. An eligibility trace achieves this via acting as variable, for each state, that decays over time and is incremented by one each time this state is re-visited, which reflects the responsibility of the frequently visited states for the returns, i.e., they have to be punished or rewarded more than the other states. Accordingly, the eligibility trace variable of each state at an arbitrary time step t is defined as follows:

Eligible states.

$$e_t(s) = \begin{cases} \gamma\lambda e_{t-1}(s) + 1 & \text{if } s = s_t \\ \gamma\lambda e_{t-1}(s) & \text{otherwise,} \end{cases} \qquad \forall s \in \mathcal{S}, \tag{12}$$

where λ represents a decay factor over time. Therefore, the approaches that involve eligibility traces have the symbol λ attached to their names in order to distinguish them, e.g., SARSA(λ). Accordingly, the update step of TD-learning (refer to Equation 11) is modified in its eligibility traces version as follows:

$$V_{t+1}(s) = V_t(s) + \alpha\delta_t e_t(s) \qquad \forall s \in \mathcal{S}, \tag{13}$$

where

$$\delta_t = r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t). \tag{14}$$

As opposed to TD learning, SARSA in its general form updates the value functions of the state-action pairs that appear in the learning episodes. Furthermore, SARSA is an on-policy approach that estimates the same policy which is used for the generation of the learning episodes, unlike off-policy methods which involve two policies,i.e., a policy that is responsible for the generation of the learning episodes and an estimation policy that is evaluated and improved. Therefore, exploration is a key factor for such approach otherwise it will not be able to learn, i.e., it has to explore various state-action pairs in order to learn from such experience because relying only on a greedy policy for selecting the actions will prevent such experience from occurring.

On-policy.

Furthermore, since SARSA considers estimating the value functions of the state-action pairs instead of those of the states, the update step takes place based on the expected return of a transition from one state to another because of an action. This

Algorithm 1 : SARSA(λ)

```
Initialize $Q(s,a)$ arbitrary $\forall s \in \mathcal{S},\ a \in \mathcal{A}(s)$;
$t \leftarrow 1$;
$e_t(s,a) \leftarrow 0 \quad \forall s \in \mathcal{S},\ a \in \mathcal{A}(s)$;
for $i \leftarrow 1$ to *number of episodes* do
    while $s_t \notin$ *terminal states* do
        $a_{t+1} \leftarrow \varepsilon$-greedy policy;
        $\delta_t \leftarrow r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t)$;
        for $s \in \mathcal{S}$ *and* $a \in \mathcal{A}(s)$ do
            $Q_{t+1}(s,a) \leftarrow Q_t(s,a) + \alpha \delta_t e_t(s,a)$;
            $e_{t+1}(s,a) \leftarrow \gamma \lambda e_t(s,a)$;
            if $s == s_{t+1}$ *and* $a == a_{t+1}$ then
                $e_{t+1}(s,a) \leftarrow e_{t+1}(s,a) + 1$;
            end
        end
        $t \leftarrow t + 1$;
    end
end
```

return is estimated based on the immediate reward from that transition in addition to the estimated state-action value of the new state while considering a new action that is chosen following our policy of interest. In other words, the update occurs relying on this sequence of events: $(s_t,\ a_t,\ r_{t+1},\ s_{t+1},\ a_{t+1})$, accordingly this approach is denoted as SARSA, i.e., following that sequence of events. Therefore, its update step compared to that of the TD learning (see to Equation 11) is defined as follows:

SARSA.

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha \Big[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \Big]. \quad (15)$$

Therefore, SARSA(λ),i.e., the eligibility traces variant of SARSA, computes the eligibility traces variables for each state-action pair as well, such that the frequently visited state-action pairs take more responsibility of the obtained returns (as opposed to Equation 12), as follows:

SARSA(λ).

$$e_t(s,a) = \begin{cases} \gamma \lambda e_{t-1}(s,a) + 1 & \text{if } s = s_t \text{ and } a = a_t \\ \gamma \lambda e_{t-1}(s,a) & \text{otherwise,} \end{cases} \quad \forall (s,a), \quad (16)$$

Therefore, the estimation of the values of the state-action pairs is modified, as opposed to SARSA (Equation 15), as follows:

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha \delta_t e_t(s,a) \quad \forall (s,a), \quad (17)$$

where

$$\delta_t = r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t). \quad (18)$$

Moreover, the steps of SARSA(λ) are illustrated in Algorithm 1.

In the next chapter, we hold a comparison between the aforementioned SARSA(λ) approach and other reinforcement learning techniques showing the advantages of SARSA(λ) over them and highlighting how these merits are urgently needed for the problems addressed in this dissertation.

3.5 COMPARISONS & DISCUSSIONS

In this section we compare SARSA(λ), i.e., which we use in the next chapters of this dissertation, to various approaches of reinforcement learning and, thus, we show why SARSA(λ) is considered the most suitable approach for the context of this dissertation. Accordingly, we briefly discuss various approaches and point out the advantages SARSA(λ) compared to them. Therefore, we focus on the main approaches of reinforcement learning which we divide into four main groups: approaches that rely on dynamic programming, Monte Carlo control techniques, temporal-learning approaches, and approaches that involve eligibility traces.

3.5.1 *Dynamic Programming Techniques*

Dynamic programming techniques tend to estimate the state value function recursively starting from an arbitrary guess about the state values. This step iteratively updates the values of all states based on the previous estimate of the value function until the value function converges. Such techniques require a model of the environment that describes the transitions between states according to the executed actions in order to evaluate the estimated policy and improve it as well. Therefore, the feasibility of such approaches are limited by the availability of the model which may be difficult in various real world scenarios.

3.5.2 *Monte Carlo Techniques*

On the other hand, the Monte Carlo approaches do not require any knowledge about the model that describes the dynamics of the learning environment. They rely on learning from samples of the state-action pairs, that appear in learning episodes that are taken from simulated or real world experiments, via averaging the returns of these state-action pairs. On the other side, they update the learned value functions after each full learning episode unlike the dynamic programming approaches at which the estimated values are updated at each time step. This behavior considers the long run effect of the selected actions, but, it results in a slow learning process.

3.5.3 *Temporal-Difference (TD) Learning*

Therefore, in order to deal with the problems of both the dynamic programming and the Monte Carlo approaches, the temporal-difference (TD) learning approach tends to combine the advantages of both of them. Similar to Monte Carlo approaches, TD learning does not require having any model about the environment's dynamics and thus it can learn from interaction with the environment via samples of state-action pairs obtained from some real world or simulated experiments. Furthermore, the learning process takes place on a step by step basis relying on the immediate rewards which leads to a faster learning process, as opposed to the Monte Carlo approaches that postpone updating the learned value functions to the end of each learning episode.

Additionally, R-Learning, which is an off-policy that relies on TD learning, focuses on undiscounted continuing learning which does not involve dividing the learning process into learning episodes. Thus, this approach does not fit the nature of the applications discussed in this dissertation which requires dividing the learning task into independent episodes to be able to train the framework on various possible situations that are represented by these episodes and, accordingly, rewarding the learning agent based on the specific terminal states reached.

3.5.4 *Eligibility Traces*

The TD learning and the Monte Carlo approaches act as extremes with respect to the rate of updating the learned value functions. The first considers only the immediate reward of the current step and estimates the rest of the return. The latter postpones the learning until the end of the learning episode. Therefore, combining TD learning with eligibility traces act as an intermediate solution between these two extremes. Applying eligibility traces is equivalent to waiting for n-steps before the update takes place and perform this update based on the immediate rewards obtained from these n-steps instead of waiting until the end of the learning episode. Therefore, combining TD learning with eligibility traces will generalize the learning performance of the TD learning and increase its efficiency.

As illustrated in Section 3.4, SARSA(λ) relies on TD learning and additionally combines eligibility traces for the frequently visited state-action pairs. Therefore, SARSA(λ) has an advantage compared to dynamic programming since it learns from direct interaction without relying on a model of the environment. Furthermore, it handles the problem of Monte Carlo techniques via providing a technique that maintains balance between the learning speed while taking the generality of the learning process and its efficiency into consideration. However, there are other approaches that belong to the same family, i.e., perform TD learning while computing the eligibility traces. Therefore, in the rest of this section, we hold a comparison between these approaches and point out why SARSA(λ), in particular, is considered the best fit for the addressed problem in this dissertation.

TD(λ), i.e., the eligibility traces extension of TD learning approach, estimates the value functions for the states, thus it is used only for the prediction of the value functions and it does not interfere with controlling the learning task (see Equation 13). On the other hand, SARSA(λ) estimates the value functions for the state-action pairs, thus it develops a control policy, as shown in Equation 17.

Q(λ) is an off-policy approach which involves two policies: a behavior policy that is responsible for the generation of the learning episodes based on which the learning via interaction takes place, and an estimation policy that is evaluated and improved. Q-learning performs TD updates for the value functions of the state-action pairs that appear during the learning episodes, i.e., the estimation policy is updated using TD independently from the behavior policy which generates the learning episodes. Accordingly, Q(λ) suffers from handling the eligibility traces for the exploration actions, which are generated from the behavior policy. For example, Watkins's Q(λ), which is one of the Q(λ) variants, cuts the eligibility traces in case an exploration action is chosen, which weakens the effect of eligibility traces in case of frequent exploration actions. On the other hand, SARSA(λ) which is an on-policy approach does not suffer from the aforementioned problems of the Q(λ) because all the actions are generated following the same policy that is estimated. Furthermore, SARSA(λ), as opposed to Q(λ), considers action selection within its update step instead of only considering the greedy action that leads to the highest expected return. Although this leads to slower learning process, it is considered a safer learning approach [Sutton and Barto, 1998].

Furthermore, we prefer SARSA(λ) compared to the Actor-Critic method with eligibility traces, which also avoids the aforementioned problems via separating the data structure of the value function from that of the policy, because SARSA(λ) was successfully deployed in a close problem [Hornung et al., 2010].

3.6 CONCLUSION

All in all, we choose to use SARSA(λ) approach for the applications presented in this dissertation, due to its ability to learn from interaction with the environment based on samples from state-action pairs without the need for a model that describes the transitions between the states and the environment's dynamics. Furthermore, it performs n-step TD prediction, i.e., it does not postpone the learning to the end of each learning episode and at the same time it does not update the values function immediately, which does not slow the learning process while generalizing its learning performance and increasing its efficiency. Moreover, it is a safe learning approach that does not cut the eligibility traces during the learning process compared to Q(λ). Finally, it is a widely used approach that have been used before in related applications.

4 FORESIGHTED PEOPLE FOLLOWING WITH KNOWN POSES

In this chapter, we present a learning model to deal with people following in assistance tasks. Our learning framework focuses on foresighted people following given known poses of both the robot and the user. Therefore, we explore the feasibility of applying a learning approach on the following problem and whether it is capable of generating navigation actions that lead to foresighted following behavior. This foresighted following behavior implies that the robot infers about the user's intended destination and does not follow the user to locations where they do not need the robot's help.

4.1 INTRODUCTION

People following with mobile robots is a challenging problem and is needed in several applications such as transportation systems in industrial settings where robots are responsible for the transportation of heavy items or dealing with hazardous substances. They can also be deployed in home scenarios, especially for taking care of elderly people as well as helping them in daily household activities. Moreover, recent studies show that interacting with robots enhances the social interaction skills of children with autism [Kim et al., 2013], since they interact better with robots compared to humans due to the simple and predictable actions of the robots. This highlights the importance of deploying such accompanying robots in applications and games that can help those children. Additionally, such following robots can be deployed in stores as autonomous shopping carts, or in scenes where robotic wheelchairs should navigate next to an accompanying pedestrian.

Mobile robots that are capable of people following are needed to assist the user in several applications.

Existing approaches have been focusing on direct following of the user via maintaining a certain distance between the robot and the user irrespectively of the user's intended destination. These approaches focus mainly on keeping the user within the robot's field of view without considering the efficiency of the robot's trajectory with respect to the traveled distance and its impact on the battery consumption and the wearing of the robot's actuators. A detailed discussion about these approaches, which demonstrate the research gaps among them, can be found in Chapter 2.

Inefficiency of direct following.

Direct following approaches suffer from inefficiency if the user does not move on the shortest path to their goal. In other words, the user may need to move to some temporary intermediate destination due to any external interrupt that may occur during the following task and in such intermediate destinations the services of the following robot may be not needed. Such situations are likely to occur in real life scenarios since humans are easily interrupted by unexpected events during navigation. For example, consider a home-assistance robot performing a delivery task and following the user when suddenly the phone rings and needs to be grabbed, or the doorbell rings. Or it might be that in the middle of the task an elderly person decides to rest for a while. In such situations, a robot that is closely following the user will have to keep following them, although the robot's assistance is not required. Thus, following the user to such destinations irrespective of the real need to the robots will lead to an inefficient behavior that affects the battery consumption. Moreover, it will lead to the wearing of the robot actuators on the long run. Therefore, it will be more efficient if the robot can infer the user's intended destination, which is initially unknown to the robot, and meets them there. However, depending on the classical ways of path prediction will not be reliable due to the uncertainties involved in such predictions due to the possibility of having more than one destination sharing parts of the paths toward them. Furthermore, the uncertainties will increase even more in case the user walks toward any unexpected intermediate destination or performs any unexpected detour. Thus, such problem needs a framework that can deal in a foresighted way while taking such uncertainties into account.

4.1.1 *Contribution*

The contribution of this chapter is to present the first solution, to the best of our knowledge, for applications involving people following tasks that consider the efficiency of the generated robot paths. We focus on scenarios in which the user does not need the robot's assistance along their way to their intended destination, but they need the robot's assistance only at that destination. In other words, the robot aims at meeting the user at their final unknown destination without the need to accompany them along their path, for example, as in delivery tasks. Therefore, it will be more efficient if the robot can infer the user's intended destination which is initially unknown for the robot and accordingly meets the user at that destination via taking shorter path than that of the user. However, as mentioned before, relying on traditional prediction techniques will not be robust due to the uncertainties involved. Thus, we propose applying reinforcement learning on the top of such prediction techniques to estimate the best navigation actions that can deal successfully with the prediction uncertainties. As we show in the experiments, our approach leads to foresighted navigation behavior that can successfully handle cases in which the user's trajectory contains detours. Our method results in paths with significantly shorter path length and significantly reduced completion time compared to direct following strategies.

Foresighted people following considers the efficiency of the following paths.

4.1.2 *Assumptions*

In this chapter, we focus on exploring the feasibility of our learning model, therefore we consider scenarios with relaxed assumptions. We assume that the poses of both the robot and the user are given with rather high accuracy. Thus, the robot does not rely on its on-board sensors in neither localizing itself nor tracking the pose of the user. Moreover, according to this assumption, there will be no possibility of occlusions. This assumption can be achieved in reality within environments that are equipped with an external motion capture system, i.e., a system of calibrated cameras that tracks the poses of a set of markers via triangularization. Additionally, we assume that the user moves in small environments in order to satisfy the assumption that these environment are equipped with external motion capture systems and to focus on exploring the capability of our framework to learn and to generalize with a bounded environments before extending that to larger ones. It is worth mentioning that these assumptions are necessary only for this chapter. On the other hand, we extend our framework in Chapter 5 to rely on the robot's on-board sensors in localization and tracking, in addition to handling occlusions and large environments.

4.1.3 *Framework Overview*

We model our problem as a Markov Decision Process (MDP) with finite state and action spaces which are bounded with the environment's dimensions and the eight possible navigation directions, respectively. To solve such MDP, we apply SARSA(λ) reinforcement learning to generate foresighted navigation actions, where SARSA(λ) is chosen since it can learn directly from interaction with the environment via simulated learning episodes without the need for an explicit model that describes the learning environment's dynamics, moreover, it provides a balanced approach compared to temporal-difference learning and Monte Carlo techniques with respect to the rate of learning updates (refer to Chapter 3 for more details). Additionally, it was previously used in rather close problems [Hornung et al., 2010]. The generated foresighted navigation actions from such framework are supposed to deal with unexpected human behaviors during the tasks, i.e., detours, such that they guide the robot to meet the user successfully at their final destination while considering the efficiency of the robot's trajectory. Our approach generates these foresighted actions via two steps, at the beginning we constantly predict the user's intended destination via a prediction model that is presented by Ziebart et al. [2009]. This model performs the predictions via a softened MDP that is independent from the one used for generation of actions. Then, based on that obtained prediction, our reinforcement learning framework infers foresighted navigation strategies that are more robust to prediction uncertainties. Following these learned navigation strategies, the robot meets the user at their intended destination efficiently, i.e., via a shorter path.

Handling predictions' uncertainties.

An overview of our framework is depicted in Figure 6. In this scenario, the user moves through the environment between different possible designated locations (as shown in the top of the figure) where it stays for a while and might need the help of a mobile robotic assistant, i.e., for general assistance tasks, social interaction, or delivery tasks. The task of the robot is to efficiently reach the initially unknown destination of the user, who might not move on the shortest trajectory. On the other hand, our learning framework generates optimal navigation actions based on predicted motions of the user (as illustrated in the block diagram at the bottom of the figure), which results in more foresighted behavior than just following the user at a close distance. Through out the learning process, we rely on

Our foresighted navigation actions are encoded in a Q-table.

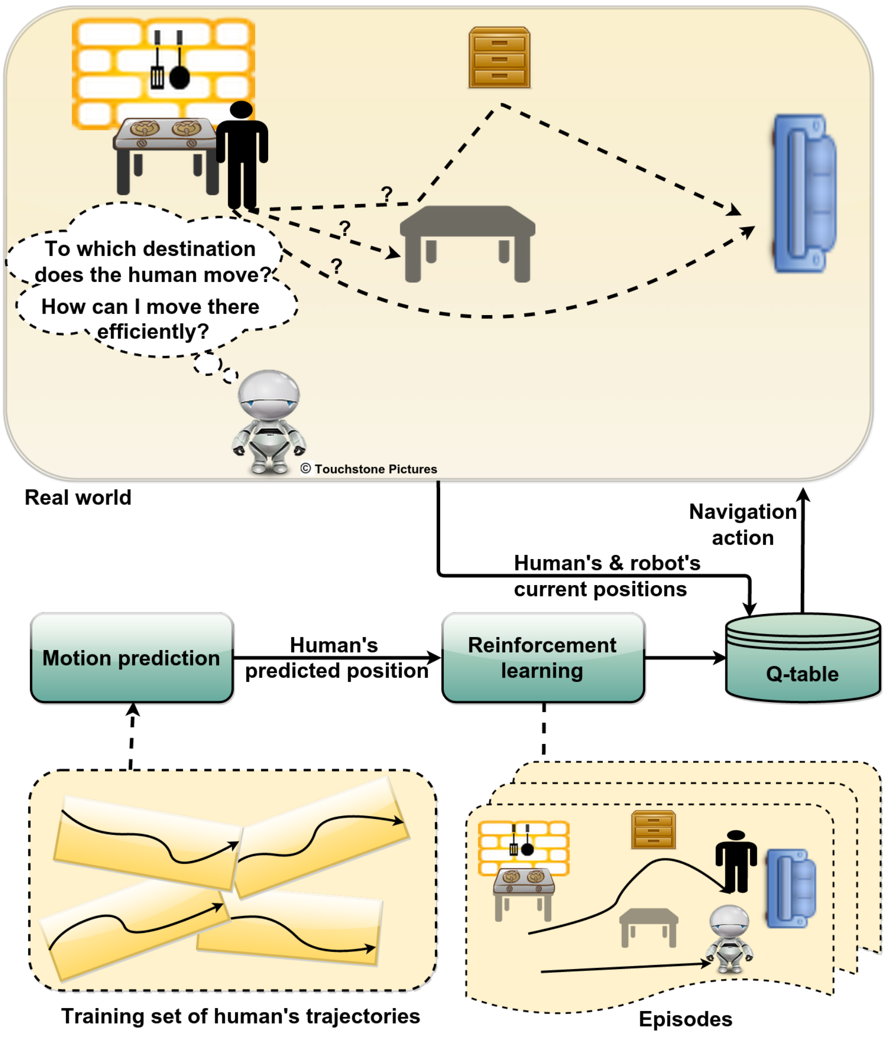

To which destination does the human move?
How can I move there efficiently?
?
?
?
© Touchstone Pictures
Real world
Navigation action
Human's & robot's current positions
Motion prediction
Human's predicted position
Reinforcement learning
Q-table
Training set of human's trajectories
Episodes

Figure 6: An overview of our learning framework.

a set of previously observed human trajectories between the possible destinations. Given these trajectories, we train the prediction model that is used to reason about the future motions and the intended destination of the user, furthermore, we use these trajectories to train our reinforcement learning framework which generates the navigation actions. The output of our learning framework is a table of Q-values, i.e., estimated values for the various state-action pairs in order to evaluate the possible actions at each state, that encodes the best navigation action for the robot based on the current robot's state.

4.1.4 *Organization*

The rest of this chapter is organized as follows: we discuss a simple prediction model that relies on a softened MDP in the next section. After that, we demonstrate our reinforcement learning framework in the form of another MDP, i.e., independent from the one used in prediction. After that, we present our findings both in simulation and real-world experiments. Finally, we draw some conclusions about the approach presented in this chapter.

4.2 MOTION PREDICTION

Motion prediction via softened MDP.

In this section, we present the motion prediction technique applied in our learning framework. We predict the user's trajectory based on the trajectory observed so far and use the prediction in the state space representation of our learning approach for generating foresighted robot actions. Our technique for motion prediction is based on the work of Ziebart et al. [2009]. This approach models the sequence of motion actions performed by a human as a softened Markov Decision Process (MDP) whose state space corresponds to the cells of a discretized grid map of the environment. The authors propose to train a prediction model using a softened version of the Bellman equation and value iteration to get a Q-table that represents the most likely motion action performed by the human at a certain position. This softened version uses the soft-maximum function instead of the ordinary maximum to be able to reason about the distribution of the trajectories, instead of considering only one single trajectory. The soft-maximum function, as defined by Ziebart et al. [2009], is as follows:

$$\mathit{softmax}_x f(x) = \log \sum\nolimits_x e^{f(x)} \tag{19}$$

and is used within the computation of the state and action values $V(s)$ and $Q(s, a)$:

$$Q(s, a) = R(s, a) + V(T(s, a)), \tag{20}$$
$$V(s) = \mathit{softmax}_a\, Q(s, a). \tag{21}$$

Here, s and a represent the user's current state and corresponding action, respectively, $T(s, a)$ is the transition function, and $R(s, a)$ is the reward after executing action a at the current state s. Equation 20 and Equation 21 are used to train the motion predictor based on a set of training trajectories with a reward function that also takes into account obstacle locations. Ziebart et al. [2009] use the obtained Q-table to predict the future destination of the trajectory observed so far as explained in the following.

Let $\zeta_{A \to B}$ denote the observed trajectory of the user from the initial state A to the current state B and $\zeta_{B \to C}$ the future trajectory from B to the unknown destination C. The probability $P(\mathit{dest}\ C|\zeta_{A \to B})$ of a certain destination C given the observed trajectory $\zeta_{A \to B}$ can then be computed with Bayes' rule, where the likelihood $P(\zeta_{A \to B}|\mathit{dest}\ C)$ intuitively depends on the ratio of the reward of $\zeta_{A \to B}$ and the expected value of $\zeta_{B \to C}$ to the value of the whole trajectory to the destination $\zeta_{A \to C}$:

$$\begin{aligned} & P(\mathit{dest}\ C|\zeta_{A \to B}) \\ & \overset{\text{Bayes'}}{=} \frac{P(\zeta_{A \to B}|\mathit{dest}\ C)P(\mathit{dest}\ C)}{P(\zeta_{A \to B})} \quad (22) \\ & = \frac{\frac{e^{R(\zeta_{A \to B})+V(B \to C)}}{e^{V(A \to C)}} P(\mathit{dest}\ C)}{\sum_D \frac{e^{R(\zeta_{A \to B})+V(B \to D)}}{e^{V(A \to D)}} P(\mathit{dest}\ D)}. \quad (23) \end{aligned}$$

Here, D corresponds to a destination among the set of possible destinations according to the training trajectories. The prior distribution $P(\mathit{dest}\ D)$ is known from the training set[1]. Furthermore, the reward of the trajectory $R(\zeta)$ is the sum of all individual rewards of state-action pairs according to ζ:

$$R(\zeta) = \sum_{(s,a) \in \zeta} R(s,a) \quad (24)$$

and $V(X \to Y)$ denotes the softmax value function of the trajectory from state X to state Y.

1 When using a motion capture system, possible destinations can be identified as places where the person frequently stays for a while without moving much. The distribution $P(\mathit{dest}\ D)$ can be learned by counting how often the person moves between possible destinations and might depend on the starting location.

We can then compute the probability of the future trajectory $\zeta_{B \to C}$ to the unknown future destination C given the so far observed trajectory $\zeta_{A \to B}$:

$$\begin{aligned} & P(\zeta_{B \to C} | \zeta_{A \to B}) \\ & \quad = P(\zeta_{B \to C} | \mathit{dest}\ C) P(\mathit{dest}\ C | \zeta_{A \to B}) && (25) \\ & \quad = \frac{e^{R(\zeta_{B \to C})}}{e^{V(B \to C)}} P(\mathit{dest}\ C | \zeta_{A \to B}) && (26) \\ & \quad = e^{R(\zeta_{B \to C}) - V(B \to C)} P(\mathit{dest}\ C | \zeta_{A \to B}). && (27) \end{aligned}$$

The term $P(\mathit{dest}\ C | \zeta_{A \to B})$ is hereby computed using Equation 23. This posterior probability is used to weight the conditional probability $P(\zeta_{B \to C} | \mathit{dest}\ C)$ of the expected future trajectory given the destination C.

In our implementation, we assume that the human can move one step at a time step in any of the eight possible directions corresponding to the neighbor cells or remain at the same state. Accordingly, our action space consists of nine actions. We set the reward to be inversely proportional to the distance from obstacles within a certain range of the human.

Thus, using the probability distribution about the future trajectory, we can predict the position of the user at a certain time step given its trajectory history. Note that since there is uncertainty in the prediction, the robot does not know the user's destination in advance and, thus, cannot directly plan a path toward it. Accordingly, we use the information about possible destinations in our navigation framework and learn optimal navigation actions as described in the following section.

4.3 LEARNING NAVIGATION ACTIONS

To model the problem as a Markov Decision Process (MDP), one needs to define a state space $\mathcal{S}$ that describes the relevant aspects of the situations that the agent may encounter and an action space $\mathcal{A}$ that represents the set of actions from which the agent can choose according to its policy $\pi : \mathcal{S} \to \mathcal{A}$. The state of the agent at a time step s_t is transformed to the new state s_{t+1} after executing the action a_t according to the transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, and the agent gets an immediate reward $r_t \in \mathcal{R}$. Then, we learn the best actions via a reinforcement learning framework, where the Q-table computed by our framework contains entries $Q(s, a)$ for each state-action pair.

4.3.1 *State Space $\mathcal{S}$*

In this chapter, we use a discretized representation of the environment in form of a grid map in which static obstacles are represented as occupied cells. The state representation of our MDP includes the relative distance between the current 2D position of the human $\mathbf{x}_t^h$ and the current 2D position of the following robot $\mathbf{x}_t^r$ at time t as well as the relative distance between $\mathbf{x}_t^r$ and the predicted position of the human $\mathbf{x}_{t+i}^h$ after i further time steps

Relative positions.

$$s_t = \begin{bmatrix} \mathbf{x}_t^h - \mathbf{x}_t^r \\ \mathbf{x}_{t+i}^h - \mathbf{x}_t^r \end{bmatrix}. \tag{28}$$

Here, we compute $\mathbf{x}_{t+i}^h$ according to Equation 27. The intuition behind choosing such a state representation is that the generated action at any time step must mainly depend on both the robot and the user's current positions. Furthermore, we add the user's predicted position in order to involve some sort of foresightedness in the generation of the actions, which involves increasing the robustness of these generated actions with respect to the prediction uncertainties. Additionally, we use relative positions instead of global positions in order to reduce the learning problem. The corresponding state space is much more compact than one that considers all possible combinations of current and predicted global positions. As shown in the experiments, this generalization works well in practice.

Robustness to prediction errors.

4.3.2 *Action Set $\mathcal{A}$ and Q-Table*

The action space consists of a set of discrete moving actions in the eight neighbor cells in addition to standing still. The Q-table computed by our learning framework contains entries $Q(s, a)$ for each state-action pair where s is defined as in Equation 28 and a is one of the nine navigation actions. Each state-action pair entry has its corresponding Q-value, which represents as an estimate to the expected return from selecting such action at that state. Thus, this Q-Table represents the intended output of our learning framework, since it is used to select the best navigation actions for the various situations of our task. We show how these Q-values are computed in Section 4.3.4.

We assume that the human moves between a set of designated locations along commonly used paths where they may stay at one of them for a while and might need the help of the robot. Therefore, one of them is the starting position of the human before moving to the next destination. Accordingly, we learn an independent Q-table for each of these designated starting locations within the map. These Q-tables are needed to be computed once for each environment. Moreover, they can be computed in parallel.

4.3.3 *Reward $\mathcal{R}$*

We designed the reward function so as to combine both the shortest path to the predicted human position as well as the difference between the distance traveled so far by the human and the traveled distance by the follower robot, as shown in Equation 29. The first term aims at guiding the robot to follow the user in a foresighted way, whereas the second term is for preferring shorter paths during the task. Thus, the learned behavior is intended to keep a balance between these two objectives. Accordingly, we define the intermediate reward r_t at time t as follows:

The reward function guides the robot in a foresighted manner via shorter paths.

$$r_t = \begin{cases} R_{max} & \text{if } t = T \\ -A^*(\mathbf{x}_t^r, \mathbf{x}_{t+i}^h) + (dist_t^h - dist_t^r) & \text{otherwise,} \end{cases} \tag{29}$$

where T is the final time step and R_{max} is a high positive reward. The function $A^*(\mathbf{x}_t^r, \mathbf{x}_{t+i}^h)$ denotes the distance resulted from applying the A* algorithm on the grid map to get the shortest path between the current robot position $\mathbf{x}_t^r$ and the predicted position of the human $\mathbf{x}_{t+i}^h$. Furthermore, $dist_t^h$ and $dist_t^r$ refer to the distance traveled until time step t by the human and the robot, respectively. The final state with time step T is reached when the robot is sufficiently close to the human (via a shorter path compared to the direct following strategy) after they have reached the next destination, which is the case when the human stays for a while close to a designated destination. Therefore, the reward function leads to the generation of foresighted navigation actions that guides the robot to meet the user at their final destination while minimizing the cost of reaching that destination.

4.3.4 *Reinforcement Learning*

We apply reinforcement learning in order to solve the aforementioned Markov Decision Process and learn the best navigation action for each possible state. We apply SARSA(λ) reinforcement learning to estimate the optimal values of the values of the state-action pairs which implies developing a policy for the robot to execute during its tasks. In Chapter 3, we have presented SARSA(λ) in detail. Therefore, we briefly point out in this section the main characteristics that fits our problem and based on which we selected this particular technique.

SARSA(λ) is an on-policy temporal-difference (TD) learning approach. Thus, it is able to learn from interaction with the environment based on a training set that allows experiencing various samples of state-action pairs without the need for a model that describes the environment's dynamics, i.e., the transitions between the states based on the possible actions. This property favors SARSA(λ) over other variants of reinforcement learning which involve dynamic programming, since they rely on having a predefined model of the environment which is not feasible in our considered application.

Furthermore, SARSA(λ) involves the use of eligibility traces, thus, it considers the effect of any action for several time steps beyond its selection at a given state, i.e., the frequently visited state-action pairs take more responsibility of the obtained returns. Therefore, SARSA(λ) is preferred compared to the TD-learning techniques that update the values function immediately and, additionally, it is better than Monte Carlo techniques which postpone the learning until the end of each learning episode. Accordingly, SARSA(λ) combines the positive traits of both techniques leading to generalizing its learning performance and increasing its efficiency, while not slowing down the learning process. Moreover, it is considered a safer learning approach compared to Q(λ), since it is an on-policy method, thus, it does not cut the eligibility traces during the learning process.

Additionally, it estimates the value function of the state-action pairs, so it develops a control policy compared to TD(λ) which estimates the value function only for the states. Furthermore, it is a widely used approach that have been used before for very close applications [Hornung et al., 2010].

Accordingly, the learning process is iterated multiple times, where each iteration is called a learning episode. A learning episode is considered complete when a terminal state is reached at time T. During each learning episode, the value of each state-action pair $Q(s, a)$, which represent an estimation of the expected return due to executing action a starting from state s following the current policy, is updated and improved, as well as the current policy that is learned so far, to approach the optimal state-action values function Q^* and the optimal policy π^*, respectively, based on the achieved immediate reward r_t at each time step t (see Equation 17 in Chapter 3).

ε-greedy policy.

Throughout the learning phase, we use an ε-greedy action selection. This selection method chooses the action with the highest Q-value with probability $1 - \varepsilon$ and all non-greedy actions with equal probability. This selection method tends to deal in a balanced way with the exploration-exploitation dilemma, in which a question is raised about whether to continue searching for better actions for a given state, i.e., to explore the state-action space more, or to exploit the so far learned pairs in order to enhance them and to maximize the obtained reward, which is unfortunately contradictory. Therefore, our selection method combines exploring various state-action pairs in addition to exploiting the Q-values learned so far via giving higher probability for selecting the greedy actions while making sure that the probability of choosing any other action is always non-zero throughout the learning process. Moreover, we decrease the value of ε after certain number of learning iterations in order to protect the learned Q-values from random distortions during proceeding with learning.

4.4 EXPERIMENTAL RESULTS

4.4.1 *Environment Setup*

In this phase of experiments we considered relatively small environments in order to focus on exploring the capability of our framework to learn and to generalize with a bounded environments before extending that to larger ones. Later in Chapter 5, we extend our framework taking into account large and complex environments. We evaluated our approach in simulated and real-world experiments in two environments each of the size of $4.8\,\text{m} \times 3.6\,\text{m}$. The first one contains three possible destinations reachable with overlapping trajectories from an initial position while the other one contains trajectories between multiple designated locations (see Figure 7). We used a map resolution of 60 cm, which yields reasonable navigation actions for the real robot with a diameter of 45 cm, however, collision checking was performed independently based on a resolution of 3 cm. Note that even if the environment consists of 48 grid cells, the size of the actual state space is much bigger since we do not only consider the relative distance between the current positions of the robot and the human but also the predicted difference in a certain number of time steps (see Equation 28).

Perfect knowledge about poses.

In this chapter, we assume perfect knowledge about the robot's and the user's trajectories. Thus, the robot does not rely on its on-board sensors in neither localizing itself nor tracking the pose of the user and therefore there is no possibility of occlusions to occur. This assumption can be realized via deploying an external motion capture system in the environment, i.e., a system of calibrated cameras that tracks the poses of a set of markers via triangularization. Moreover, we assume that both the robot and the user are moving approximately with the same velocity, which only affects the prediction about the user's future location.

4.4.2 *Trajectory Generation for Training and Testing*

We randomly generated human trajectories for both, the training and test phases. However, a possible future extension is to use real data acquired either from motion capture systems or from 2D videos of human activity via trajectory extraction techniques [Boukhers, 2017]. These randomly generated trajectories are composed of straight line segments of 25 cm length. The orientation of the segments relative to the destination is chosen uniformly from the interval $[0°, 60°]$. For the map in Figure 7a, our training set consists of 60 trajectories, 20 for each of the three possible destinations. For the map in Figure 7b, we generated a training set consisting of 15 trajectories for each of the possible combinations between the designated locations. For testing, we used five randomly generated trajectories for each motion class. It is worth mentioning that the prediction of destinations with overlapping trajectories is more difficult and leads to uncertainties regarding such predictions, thus, relying only on these predictions and performing path planning based on them will lead to inefficient robot trajectories.

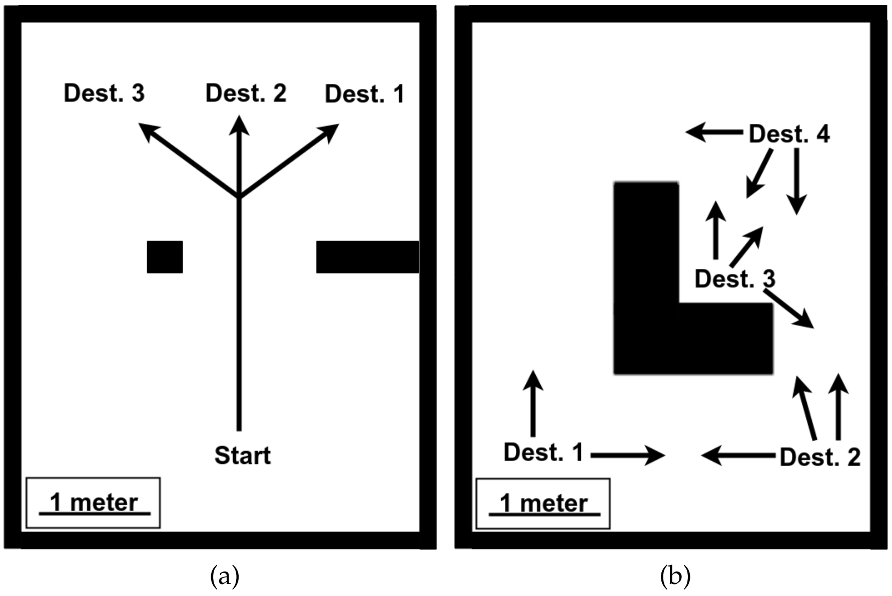


Figure 7: Maps used for the experiments: (a) Environment with a fixed start position and three possible destinations with overlapping trajectories, making the prediction more difficult. (b) Environment with several designated locations between which the human moves.

4.4.3 *Parameters*

Exploration vs exploitation.

During the learning of the Q-table, we used an initial value of 0.4 for ε of the greedy action selection to allow for exploring the state space and decreased the value to 0.2 after a certain number of iterations. This leads to more exploration of the state and action spaces at beginning of the learning process compared to more exploitation of the learned Q-values after a considerable leaning iterations. For the execution, we used a value of 0.05, i.e., the robot chooses the action with the highest Q-value with probability 0.95. The Q-tables for our experiments were learned from a minimum of $12,000$ learning episodes. We consider a learning episode as successful if the distance of the robot to the human destination is smaller than 1.2 m within a maximum number of 100 time steps, otherwise the episode is aborted. In the successful learning episode, we apply a high positive immediate reward R_{max} with a value of $10,000$ at the terminal time step (see Equation 29). We experimentally determined the value of 3 for i for the prediction in Equation 28 and Equation 29 to work best with the chosen map resolution of 60 cm.

In the learning episodes and test runs, we generated the robot's starting position randomly within a range of 60 cm around the human's initial position. In the test runs, we abort the execution in case the robot does not reach the user's destination within 25 time steps after the user arrives there, however, in real applications the robot can drive to the last predicted location of the user as we show in the extension of our work in Chapter 5.

4.4.4 *Evaluation Metrics*

Efficiency of following paths.

The main goal of this approach is to generate more efficient robot trajectories compared to the direct following strategies. Therefore, in order to evaluate our approach, we computed the saving with respect to the path length by comparing the distance traveled by the robot according to our learned navigation actions to that of applying a direct following method, in which the robot follows the shortest path to the position of the user at each time step. Furthermore, since the poses of both the robot and the user are assumed to be known at every time step, it is necessary to evaluate the time needed by the robot to meet the user successfully at the destination. Thus, we evaluated the completion time of our approach to a 'wait-and-observe-first' strategy in which the robot waits until the user reaches its destination and only then starts moving according to the shortest path to the destination. To show statistical significance of our results, we applied a *statistical two-tailed paired t-test* with a confidence interval of at least 95%.

4.4.5 *Experiments in Simulation*

We performed 250 runs for each test trajectory to alleviate the randomness arising from the generation of the initial position of the robot as well as the randomness involved in the generation of the actions. As can be seen from Table 1 (first and third row, first column), our approach significantly outperforms the direct following strategy with respect to the traveled distance in both environments. Note that in the environment of Figure 7b, in some cases the robot can directly predict the correct navigation goal after one step and immediately move there. Depending on the user's and the robot's start location, the robot then reaches this destination faster than the user.

Furthermore, we added a number of non goal-directed trajectories (i.e., trajectories that involve detours) to the test set (25%) of the environment in Figure 7a to show the effectiveness of the robot behavior generated by our learning framework and its ability to handle unexpected trajectories. These detours represent unexpected behaviors that the user may perform due to the occurrence of external events that may interrupt their path to their destination, such as grabbing a phone that suddenly rings, responding to the doorbell ring, or even the need to have some rest in the case of elderly people.

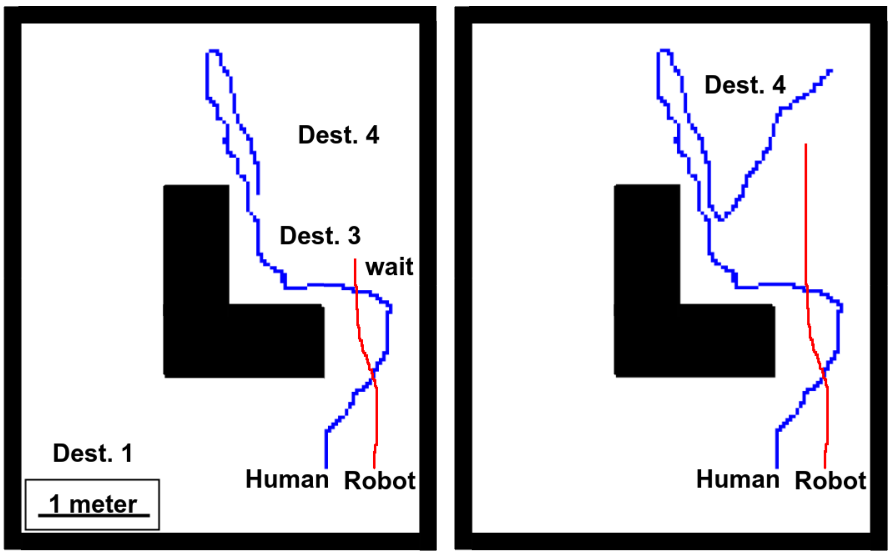
Detours.

These non goal-directed trajectories were not included in the training phase and the generation of the Q-table for the navigation actions to avoid the over-fitting of the training for such scenarios. In the corresponding runs, the user trajectory leads around the obstacle in the left part of the map (similar to the user's trajectory in Figure 10). This can be seen as the case where the user moves to fetch some items and then continues walking to its actual destination. In such scenarios, our foresighted following robot waits for the user until they are back from their detour then the robot resumes the following task. If such scenarios are included in the test set we even achieve an overall average gain of 19.1% (see Table 1 second row, first column).

When comparing our approach to the 'wait-and-observe-first' strategy regarding completion time, we achieve an average gain of 13.1% - 14.6% (Table 1 second column). Furthermore, all the reported results in Table 1 pass a *statistical two-tailed paired t-test* with a confidence interval of at least 95%.

Table 1: Gain in traveled distance and completion time

Test trajectories of environments		**Gain in distance**	**Gain in time**	**Statistically significant ($\alpha = 0.05$)**	
				Dist.	**Time**
Figure 7a	Basic set	7.9%	13.1%	Yes	Yes
	With cycles	19.1%	14.6%	Yes	Yes
Figure 7b Basic set		18.3%	14.2%	Yes	Yes



(a) (b)

Figure 8: Experiment in which the user first makes a large detour before walking toward their intended destination. (a) The robot does not follow the user but waits first. (b) Only as the user continues moving in the direction of the actual destination, the robot follows and, thus, avoids the detour that would have resulted from closely following the user.

In 8% of the test runs, the robot was caught in local minima and did not reach the destination within a fixed time limit while choosing actions according to the Q-table. The corresponding runs are not included in the evaluation. Note that we applied only the Q-table for action selection in our approach and did not combine it with moving on the shortest path to the actual destination after the user arrived there, which would be reasonable in practice and improve the performance in some cases, in particular when facing the local minimum problem mentioned above.

We performed an additional experiment to illustrate the strength of our approach, where this scenario was not also included in the training phase. Here, the user was only walking to their final destination after performing a larger detour (see Figure 8a). This unexpected behavior was correctly handled by the robot as it did not follow the user but waited. Only as the user continued their path toward the destination, the robot started moving again (see Figure 8b). Thus, the robot behaves in a foresighted way and could avoid the large detour that would have resulted from closely following the user.

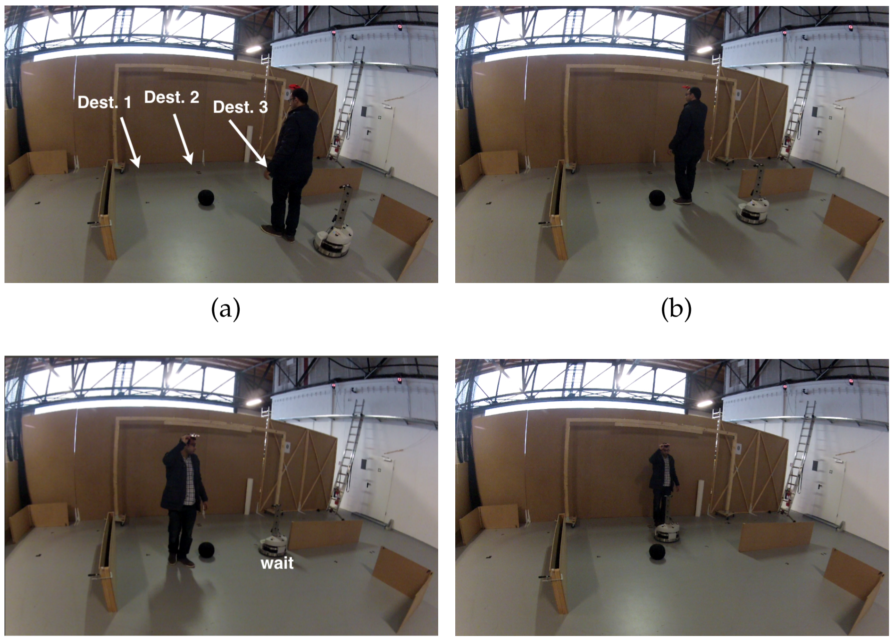


Figure 9: (a) The user starts walking toward his (unknown) destination and the robot executes navigation actions to follow him. (b) The user does not move directly to his destination but walks in a cycle around an object. (c) The robot behaving according to our learned policy ignores this cycle and waits. (d) The user continues walking toward his destination, followed by the robot.

4.4.6 *Experiments with a Real Robot*

We also carried out experiments with a real robot (Robotino by Festo) to test the learned navigation strategy. To detect the position of the user and localize the robot, we used an external motion capture (MoCap) system and retroreflective markers attached to both the user and the robot. We focused on the scenario where the trajectory of the user does not directly lead to his destination. As shown in Figure 9, the user performed an inefficient trajectory by walking around the obstacle in the left part instead of directly going to his goal location, where this scenario was not included in the training phase. As can be seen, the follower robot ignored this cycle and waited instead in a foresighted manner before proceeding to follow the user afterwards to its intended navigation goal. The corresponding path (see Figure 10) is much shorter than the user's trajectory.

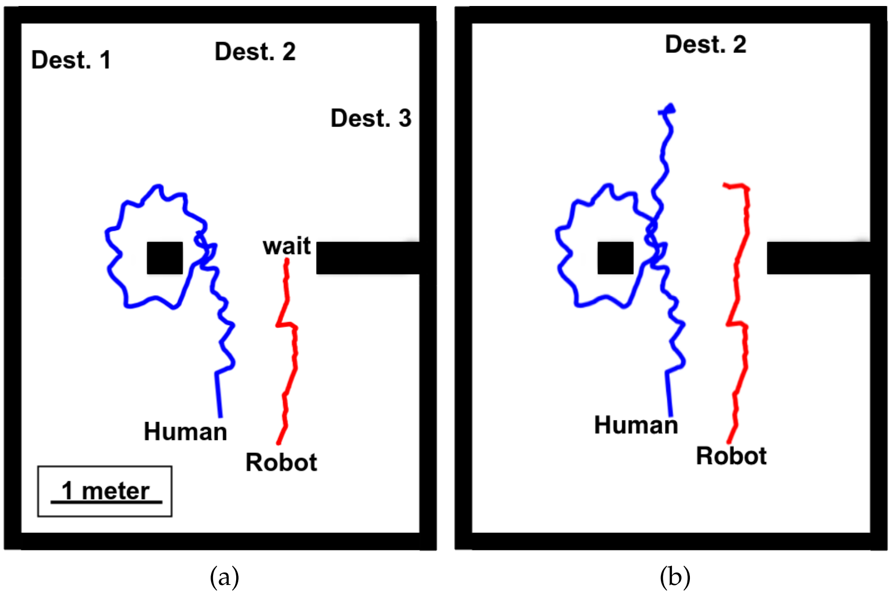


Figure 10: Trajectories of the human and the robot corresponding to the experiment shown in Figure 9 recorded by the MoCap system. (a) The robot's trajectory is rather efficient, as the robot correctly predicts that the human will continue moving to one of the destinations in the area on top and therefore waits for the human half way. (b) Both human and robot resume the path to the destination.

4.5 CONCLUSIONS

In this chapter, we demonstrated an approach that generates foresighted navigation behavior of an assistance robot. We consider the scenario in which the human moves between different designated locations where they might interact with the robot. Thus, the task of the robot is to reach these places while minimizing trajectory length and completion time. Our framework relies on a learned prediction model for the human's motions that is used to reason about its future trajectory and target destination. Based on that prediction and the current robot position, we apply reinforcement learning to generate foresighted navigation actions for the robot. These generated foresighted actions are robust against the prediction uncertainties as well as the unexpected behaviors and detors of the user. As the experiments carried out in simulation and with a real mobile robot demonstrate, our approach leads to foresighted navigation behavior resulting in significantly shorter paths and a significantly reduced completion time compared to naive following strategies.

In the next chapter, we extend our system to use the robot's on-board sensors for people tracking, which will lead to occlusions and uncertainty about the human's position. So the robot will also need to learn when to consider active re-localization of the human based on its predicted motions. Furthermore, we extend the state-action space as well as the reward function to handle such occluded situations. Moreover, we consider larger and more complex environments that demonstrate the strength of our approach in various situations. Additionally, we replace the prediction model with a more robust one.

5 FORESIGHTED PEOPLE FOLLOWING UNDER OCCLUSIONS

In this chapter, we focus on attaining a performance that better fits real world scenarios in which the robot is vulnerable to occlusions due to its limited field of view as well as obstacles that restrict this limited field of view even more. Therefore, we extend our people following framework that was discussed in the previous chapter such that the robot relies on its sensors to localize itself and to track the user.

5.1 INTRODUCTION

In this chapter, we consider scenarios in which a service robot needs to encounter a user at designated locations, between which the human moves along commonly used paths. One example application are transportation tasks where the robot has to provide items to the human at predefined places. Such transportation tasks can be vital in case of transporting hazardous materials and chemicals, or heavy items. Additionally, such robots can be deployed as autonomous shopping carts in stores or in massive warehouses. Furthermore, robots with such capabilities can be used in the treatment programs of the children with autism via designing some interactive games to enhance the children's interactive capabilities and social skills. Since the interaction of children with autism with robots proved better results compared to adults [Kim et al., 2013].

Example applications of people following tasks.

Following the human at a certain distance might enable the robot to solve such tasks and various techniques exist to closely follow people via moving the robot on the shortest path to the user at each time step [Harmati and Skrzypczyk, 2009; J. Huang et al., 2006; L. Huang, 2009; Kuderer and Burgard, 2014; Nascimento et al., 2013; Pradhan et al., 2013], refer to Chapter 2 for more details. However, since humans may not always take the shortest path to their next destination, i.e., due to unexpected external interrupts that may occur such as a phone rings and needs to be picked up, such methods that rely on direct following of the user might lead to an inefficient robot navigation behavior in the considered scenarios where the robot has to interact with the user *only* at designated places. Additionally, robots that rely on on-board sensors to localize and track the user may face some problems due to possible occlusions that may occur. Furthermore, the user might occasionally move through passages that are impassable to the robot, i.e., due to size or safety constraints, such that the robot is forced to find an alternative route. Accordingly, direct following techniques will face serious problems in the aforementioned situations. In such cases the problem arises that the user will eventually be out of the robot's field of view, which will lead to an uncertain estimate about the user's location and a wrongly predicted destination. Such techniques will get stuck in case of occlusions, moreover, they will not be able to find alternative paths to those impassable by the robot and will get stuck as well. Thus, the robot will need to consider active re-localization of the user to improve the estimate and to be able to infer the next navigation goal.

Occlusions are critical for direct following.

5.1.1 *Contributions*

In this chapter, we reduce our previously mentioned assumptions in Chapter 4 in order to mimic real world scenarios, thus, we rely on the on-board sensors of the robot in localizing the user instead of our previous assumption of having perfect knowledge about the user's pose. Therefore, we use a particle filter-based predicted model in order to retain multi-modal hypothesis about the user's location in case of occlusions to have more robust predictions. Moreover, we modify our previously discussed approach to be able to handle such occlusions via modifying the state space and the reward function to generate foresighted navigation action that aims at keeping balance between having the user in the robot's field of view and minimizing navigation costs. Furthermore, we consider large complex environments, to highlight the strength of our approach, without overloading the learning process.

Foresighted following using on-board sensors.

5.1.2 *Framework Overview*

An overview of our proposed system is shown in Figure 11. Initially, we use the motion prediction model to learn navigation strategies via reinforcement learning. The output is a Q-table that provides the robot with the best navigation action for the current situation. Concerning the motion prediction, we apply an approach to constantly predict the human's position at future time steps based on the robot's previous observations of the human's position. As opposed to our previous approach presented in Chapter 4, we use a particle filter-based predicted model that can handle multi-modal hypothesis about the user's likely locations in case of occlusions. However, we can not rely solely on such prediction due to uncertainties caused by the occlusions. Thus, similar to Chapter 4, this prediction is used as an input to a reinforcement learning framework to generate foresighted robot navigation actions that are robust to such uncertainties.

We pass the predictions to the reinforcement learning framework to generate robust actions to prediction errors.

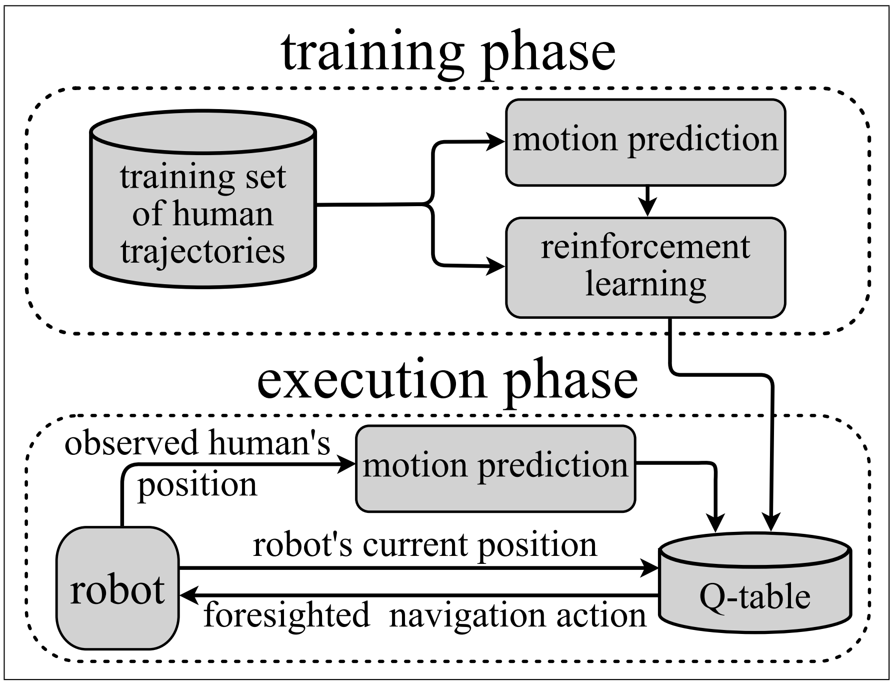


Figure 11: Based on the robot's current position and its observations about the human, the robot chooses a navigation action using the foresighted navigation strategy that was learned off-line from a training set of observed human trajectories.

Regarding the learning of navigation actions, similar to Chapter 4, we also model our problem as a Markov decision process (MDP) and thus we rely on SARSA(λ) reinforcement learning to solve such MDP due to its ability to learn from direct interaction with the environment via simulated learning episodes without the need for a model that describes the environment's dynamics. Furthermore, SARSA(λ) is a balanced approach between the speed of the learning and its safety, refer to Chapter 3 for more details. In addition to that, SARSA(λ) was proven to be able to learn good navigation actions in Chapter 4. However, we modify the map representation as well as the state space representation to be able to handle large environments and possible occlusions. Furthermore, we enhance the reward function in order to maintain balance between having the user in the robot's field of view and minimizing navigation costs, in addition to handling occlusions and finding alternative paths to those that are impassable by the robot. Additionally, we allow the robot to explicitly perform observation actions to update the belief about the user's location via autonomously selecting key locations at which the robot waits and observes, where these locations have distinguishable visibility characteristics (ex: intersection points, corridors ... etc) which increases the likelihood of observing the user again. The resulting navigation strategy leads to efficient robot paths and observation actions that improve the prediction of future human movements.

Figure 12 highlights the strengths of our approach. The user initially moves along a trajectory that might correspond to two different paths, while the robot is only needed if the human moved along path 2. Our modified approach leads to an efficient strategy where the robot follows the human to the place where the paths diverge and observes the future behavior of the user, hence, reducing unnecessary movement actions. Following the user at close distance would also be a solution to reduce uncertainty, but this often results in unnecessary detours and interferences with the human.

Extensions compared to the previous chapter.

All in all, we extended our previous work presented in Chapter 4 by using a more compact representation of the movement possibilities, implementing a particle filter based prediction, modifying the state space, enhancing the reward function to deal with occlusions, and allowing the robot to explicitly perform observation actions to update the belief about the human's location. Moreover, as we show in simulated and real-world experiments, our framework generates foresighted navi-

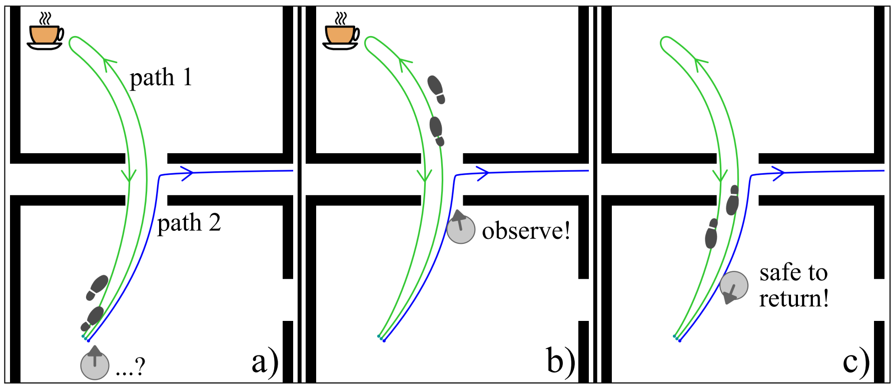


Figure 12: Example of foresighted people following: In a scenario where a human commonly walks on different known paths, our approach enables the robot to act efficiently. a) When the human starts walking the robot does not yet know which of the two paths the human will take and simply follows. b) The robot predicts the path of the human and remains at a spot, where possible deviations from the prediction are well visible. c) The robot sees the human return and immediately drives back to the initial spot.

gation actions that can also deal with cases in which the human is not in the robot's field of view. The trajectories generated by our approach are significantly shorter compared to a direct-following approach and we achieve a higher number of successful runs. Furthermore, we compare the performance of our approach to that of a heuristic that depends only on predicting the user's future location and moves the robot toward it. We show in this comparison how such heuristic suffers from prediction uncertainties and on the other side how our approach is able to overcome such uncertainties and generate foresighted navigation actions that lead to efficient robot trajectories.

5.1.3 *Organization*

The rest of this chapter is organized as follows: we state the addressed problem of this chapter in the next section. Then, we discuss a particle filter based prediction model to predict the human's motion. After that, we demonstrate our modified reinforcement learning framework. After that, we present our findings both in simulation and real-world experiments. Finally, we draw some conclusions about the approach presented in this chapter.

5.2 PROBLEM FORMULATION

We consider a scenario where the robot knows the structure and static obstacles in the environment. We assume that the human commonly walks on typical paths between different points of interest (destinations) where a robot is needed to assist the human. Our aim here is to exploit the given information in order to generate more efficient navigation actions for the robot compared to a direct-following method and additionally deal with situations where such a method would fail. An example of the former is discussed in Figure 12. Furthermore, since it sometimes happens that the human moves along paths which are impassable for the robot, the robot should learn in these cases how to move to the predicted destination of the human, thereby performing explicit observation actions to update the prediction of the human position if useful. All in all, the aim of our approach is to reach the correct destination of the human while reducing the overall path length.

5.3 MOTION PREDICTION

Topo-metric graph representation.

To track the user while moving along paths between a set of predefined destinations, we use a particle filter following ideas of the work by Liao et al. [2003]. We restrict the propagation of the particles over a topo-metric graph that represents the environment (see Figure 13). The particles are propagated, over the nodes of the topo-metric graph, according to the user's velocity, following the common paths that the user may take. The poses of the particles are initialized at the graph node that corresponds to the user's known starting location. Each particle moves on the graph toward one of the destinations, reachable from the previous destination (the user's starting location).

Particles' propagation.

For each particle we independently sample one of the possible destinations based on transition probabilities between the destinations. Then, at every time step, each particle moves to a graph node along the path to its current destination according to a Gaussian motion model based on the estimated velocity of the user, i.e., each particle may move to a node along the path, stay at the current node, or move to other nodes that do not belong to the intended path following a Gaussian motion model. Such motion model is necessary to be able to handle unexpected behaviors of the user, i.e., if the user does not follow one of the common paths and performed some detour in order to pick some items before resuming their path to one of the predefined destinations.

The weights of the particles correspond to the observation likelihood. High weights are assigned to the particles that are close to the user's observed position. In case the user is not observed, the weights of the particles in the field of view are reduced, thus, our belief about the user's existence at other locations increase. Therefore, observation actions are vital and lead to information gain that updates our multi-modal belief of the user's location whether the user is within the robot's field of view or not.

Weights are proportional to the observation likelihood.

Furthermore, at every time step, we perform a selective resampling to enhance the particles that correspond to good beliefs while avoiding the particle depletion problem, i.e., the problem of performing unnecessary resampling which leads to losing important particles. Thus, we compute the number of effective particles according to the updated normalized weights, i.e., which corresponds the inverse variance of these weights, and if the number of effective particles is below half the total number of particles we perform a low-variance resampling. In other words, we only perform resampling when the current distribution of particles is meaningless and far from the intended distribution according to the updated weights.

To predict the user's future location, we simulate the propagation of the particles according to the motion model described above a few time steps into the future and then determine the predicted graph node based on the cluster of particles that has the highest weight. According to the aforementioned approach, the prediction model can correct itself via observations, i.e., either positive or negative, and thus it leads to more robust predictions. However, we can not rely solely on the generated prediction and perform path planning based on them, because these corrections will be on the cost of the efficiency of the robot's trajectory, i.e., the robot still needs to move toward some initial predictions which are not guaranteed to be correct due to the involved uncertainties and during this process the prediction model corrects itself. Therefore, we need to generate foresighted navigation actions that are able to overcome such uncertainties, which is illustrated in the upcoming section.

Particle filter based predictions.

5.4 LEARNING NAVIGATION ACTIONS

In order to generate actions for the robot while observing and following the human, we model the problem as a Markov decision process (MDP), and combine it with the human motion prediction (see Section 5.3). We apply reinforcement learning on such MDP to learn a Q-table that represents the navigation policy of the robot. For every possible starting location of the human (corresponding to the points of interest/destinations), we learn a separate Q-table that encodes the suitable policy for the set of the user's common paths which start from that location.

The remainder of this section explains in detail all aspects of our learning approach. First, we define a state space $\mathcal{S}$ that describes the relevant aspects of the situations that the agent may encounter and an action space $\mathcal{A}$ that represents the set of actions from which the agent can choose according to its policy $\pi : \mathcal{S} \to \mathcal{A}$. Then, we define our modified reward function that evaluates the chosen actions and their effect on the states' transitions, i.e., the state of the agent at a time step s_t is transformed to the new state s_{t+1} after executing the action a_t according to the transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, and the agent gets an immediate reward $r_t \in \mathcal{R}$. Then, we discuss the SARSA(λ) reinforcement learning approach based on which we learn the best actions in the form of a Q-table that contains entries $Q(s, a)$ for each state-action pair.

5.4.1 *State Space $\mathcal{S}$*

As mentioned in Section 5.3, for the prediction of the human's motion, we represent the environment as a discretized grid map with an overlaid topo-metric graph. For the state space and the action generation we use the same representation, such that every grid cell is mapped onto the closest graph node within the same room. Additionally, every graph node is mapped onto the grid cell, which lies at the same position as the node. The topo-metric representation is a compact representation of the movement possibilities in the environment and scales better with larger maps. The corresponding map representation as topo-metric graph for our quantitative simulation experiments is depicted in Figure 13. Note that the environment contains a

More concise representation.

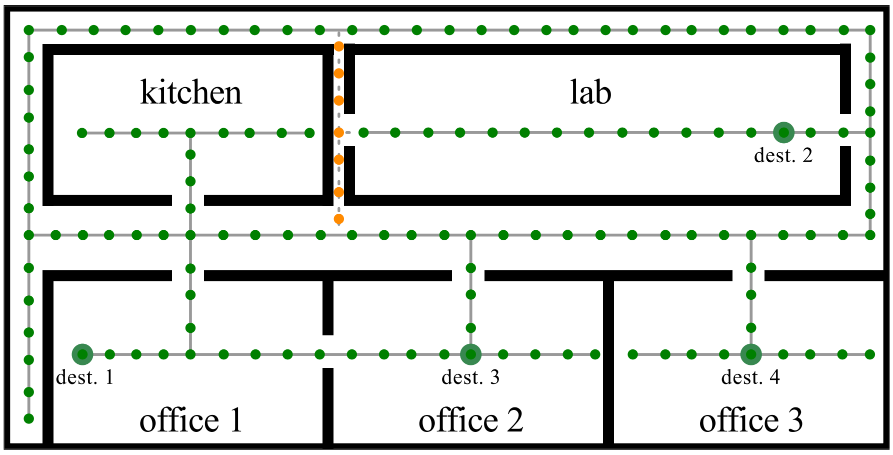


Figure 13: Simulation environment with topo-metric graph. The environment is represented as a grid map with an overlaid topo-metric graph, such that each grid cell corresponds to the nearest graph node (green dots) within the same room. At the same time, each graph node is mapped onto the grid cell, which lies at the same position. The orange dots correspond to nodes that are only passable by the human but not by the robot.

passage that is not passable by the robot (orange nodes) and, thus, the corresponding navigation actions are excluded.

The state space includes the current graph node position of the robot $\mathbf{x}_t^r$ at time t and the predicted position of the human, also mapped onto the graph, $\mathbf{x}_{t+i}^h$ after i future time steps:

$$s_t = \begin{bmatrix} \mathbf{x}_t^r \\ \mathbf{x}_{t+i}^h \end{bmatrix}. \tag{30}$$

Due to the compact representation of the environment as a topo-metric graph, this state space is much more concise compared to our previous work that relies on the 2D relative positions of the robot's current location with respect to both the user's current location as well as their predicted location (see Section 4.3.1 in Chapter 4 for more details). Such concise state space is important since it leads to a faster learning process. Furthermore, we do not include the human's current position here since it does not give more information and, apart from that, might not be observable. Therefore, in contrast to our previous state representation, the new formulation can scale for reasonably large environments in addition to its ability to handle occlusions.

The topo-metric graph does naturally not cover the whole grid map and the real robot might not perfectly drive along the graph. Therefore, the graph node position of the robot is always determined according to the robot's current grid cell mapped onto the graph, similarly as for the human.

5.4.2 *Action Set $\mathcal{A}$ and Q-tables*

The action space consists of movement actions, a waiting action, and an observation action. Movement actions are defined along the graph and consist of movements to all neighboring nodes. Therefore, the amount of movement actions depends on the current node position of the robot, as certain nodes have more neighbors compared to others, e.g., intersections. An observation action consists of a full 360° rotation or a rotation until the human is observed. Finally, successive waiting actions are converted to observation actions if the user is not in the field of view while the robot is waiting. This approach helps to keep the human in the field of view of the robot while the robot is waiting, e.g., in the case of long detours of the user.

In order to execute actions in the real world we need to map the action space, i.e., movements between neighboring graph nodes, onto real actions of the robot. We accomplish this by determining the corresponding grid cell position of the graph node after performing an action and sending the real 2D coordinate as a new local goal to the robot.

5.4.3 *Reward Function $\mathcal{R}$*

During the training phase, we define the immediate reward r_t at time t as follows:

$$r_t = \begin{cases} R_{max} & \text{if } t = T \\ -A^*(\mathbf{x}^r_t, \mathbf{x}^h_D) - \nu \cdot A^*(\mathbf{x}^r_t, \mathbf{x}^h_{t+i}) - C & \text{otherwise} \end{cases}, \qquad (31)$$

where T is the final time step in the case of a successful learning episode and and R_{max} is a high positive immediate reward. A learning episode is considered as successful if the robot reaches the human's destination through a shorter path compared to an approach that uses the concept of directly following. This direct following approach also moves along the graph, however, at every time step simply moves one node toward the human's current position.

The function $A^*(\mathbf{x}_t^r, \mathbf{x}_D^h)$ in Equation 31 denotes the distance resulting from applying the A* algorithm to get the shortest path from the robot's current position $\mathbf{x}_t^r$ to the human's intended destination $\mathbf{x}_D^h$, which is known from the training data. The term $A^*(\mathbf{x}_t^r, \mathbf{x}_{t+i}^h)$ represents the A* distance between the robot's current position and the predicted position of the human after i time steps, while ν is 0 in case the human is currently visible and 1 otherwise. Finally, C represents the cost for performing an action, which is 0 in case of waiting and observation actions and the distance between two neighboring nodes in case of a movement action.

Motivation of the reward function.

The first term of Equation 31 guides the robot toward the goal while the second term helps to keep the human in the robot's field of view. The last term minimizes the total amount of movement actions, which is desirable if the human takes detours or moves to areas where the robot is not required.

Our previous work presented in Chapter 4 can not deal with occlusions since it assumes perfect knowledge about the user's pose (see Section 4.3.3 for more details). On the other hand, our modified reward function in Equation 31 is more general and realistic since it maintains balance between different aspects. It guides the robot to the correct final destination as well as trying to keep the user in the robot's field of view as much as possible while considering the efficiency of the robot trajectory. Thus, the generated actions are foresighted actions that are able to make sure of the visibility characteristics of the environments, i.e., via autonomously choosing key locations to perform the observation actions.

5.4.4 *Reinforcement Learning*

In order to solve the aforementioned Markov decision process (MDP), similar to our previous work in Chapter 4, we apply SARSA(λ) reinforcement learning in order to learn the best navigation action for each possible state. Mainly, we choose to continue using SARSA(λ) in this chapter due to its successful performance which we have presented in Chapter 4. Additionally, in this section, we briefly present a quick reminder about the main characteristics based on which we chose to apply SARSA(λ) in the first place, however, a detailed discussion can be found in Chapter 3.

We choose SARSA(λ) because it does not require any modeling of the environment, however, we can train it using sample examples of the user's trajectories. We denote these simulated experiments that are used for training as learning episodes. Furthermore, SARSA(λ) is considered as a balanced approach between the Monte Carlo techniques and the temporal-difference (TD) learning approaches. A main drawback of Monte Carlo techniques is postponing the learning to the end of each learning episode and, on the other side, the TD learning approaches update the values function immediately at each time step which does not consider the effect of the chosen actions on the long run which degrades the generality of the learned policy. However, SARSA(λ) waits for n-steps before updating the values via the use of eligibility traces that are memory variables that keep track of the frequently visited state-action pairs and, thus, they reward the responsible actions for their subsequent effects without postponing that until the end of the learning episode.

As mentioned above, the learning process is iterative where each iteration is called a learning episode, where a learning episode terminates successfully when the robot meets the human at their final destination at time T via a shorter path compared to direct following techniques. In each learning episode the Q-value $Q(s, a)$, which represent the expected return of executing action a at state s following the current policy, is updated for each state-action pair that belongs to this episode (see Equation 17 in Chapter 3). Such updates are reflected on updating the so far learned policy.

ε-greedy policy.

The learned action policy should aim at maximizing the return of each state, which corresponds to choosing the action with the maximum Q-value. However, since the robot may get stuck in a local-minimum at a certain graph node, we apply ε-greedy action selection to allow more exploration of the state-action search space. This policy selects the action with the maximum Q-value with probability $1 - \varepsilon$ or any of the other non-greedy actions randomly with probability ε. Thus, this selection method tends to deal in a balanced way with the exploration-exploitation dilemma via always having a non-zero probability for choosing any possible action.

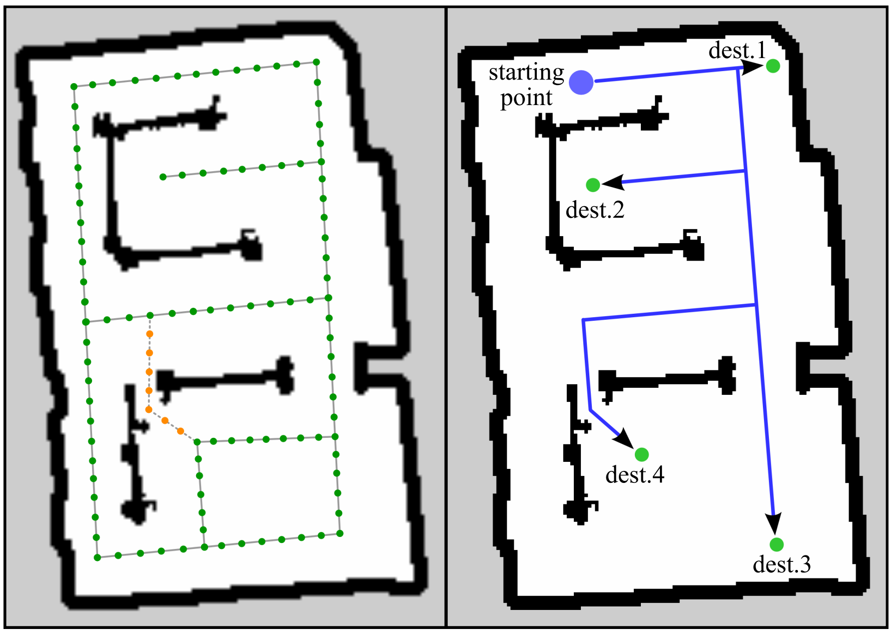


Figure 14: Left: The real-world environment with an overlaid topometric graph. The orange dots correspond to nodes that are only passable by the human but not by the robot. Right: Common human trajectories used in the real-world environment.

5.5 EXPERIMENTS

We carried out both simulated and real-world experiments to evaluate our proposed approach and compare it to the direct method approach. In Section 5.5.1, we demonstrate the setup of our experiments. Then, we discuss the achieved results of the simulated and real-world experiments in Section 5.5.2 and Section 5.5.3, respectively.

5.5.1 *Experimental Setup*

We evaluated our approach, both in simulated and real-world experiments. The size of the simulation environment is 41 m×20.5 m with a map resolution of 0.25 m and a map of size 4.8 m×7.6 m with a resolution of 0.05 m for the real-world experiment (see Figure 13 and Figure 14, respectively). The distances between the graph nodes in the simulated and real-world environments are 1.5 m and 0.5 m, respectively. To show one strength of our approach, each of the environments contains a narrow passage that is not passable by the robot due to size constraints. Thus, directly following the human along these passages is not possible and will not succeed, while our approach tries to predict the destination of the human and learns how to best move in order to encounter the human at the destination, thereby considering also observation actions along the way. Note that the heuristic strategy of just moving to the predicted destination on the shortest path does not consider observation actions and will sometimes be affected by wrong predictions.

Negative observations. In case the human is outside the robot's field of view, we propagate the particles and update their weights, i.e., reduce their weights if they are within the robot's field of view, and then determine the predicted graph node based on the cluster of particles that has the highest weight. Therefore, performing observation actions at locations that have distinguishable visibility characteristics (ex: intersection points, corridors ... etc) is vital, moreover, this is determined autonomously by our framework via the learned actions.

Parameters. During the reinforcement learning phase, we initialize ε of the greedy action selection policy with 0.35 to allow for a wider exploration of the state and action space. During the execution phase, the robot follows the learned policy represented by the Q-table, i.e, $\varepsilon = 0$. Each Q-table is learned in simulation from a maximum of 4000 learning episodes with a maximum of 100 time steps. In the successful learning episode, we apply a high positive immediate reward R_{max} with a value of $10,000$ at the terminal time step (see Equation 31). We experimentally determined that a three-step prediction ahead, i.e., $i = 3$ in Equation 30 works best.

5.5.2 *Experiments in Simulation*

For the simulation experiments we simulated 105 human trajectories, 15 for each possible route (see Figure 15). 60% of the trajectories were used for the training and the rest for the evaluation of the learned robot behavior. As illustrated in Figure 15, these common passes cover various scenarios, i.e., some of them perform unnecessary detours and others correspond to shortest paths between destinations, in addition of considering a path that is impassable by the robot due to size constraints.

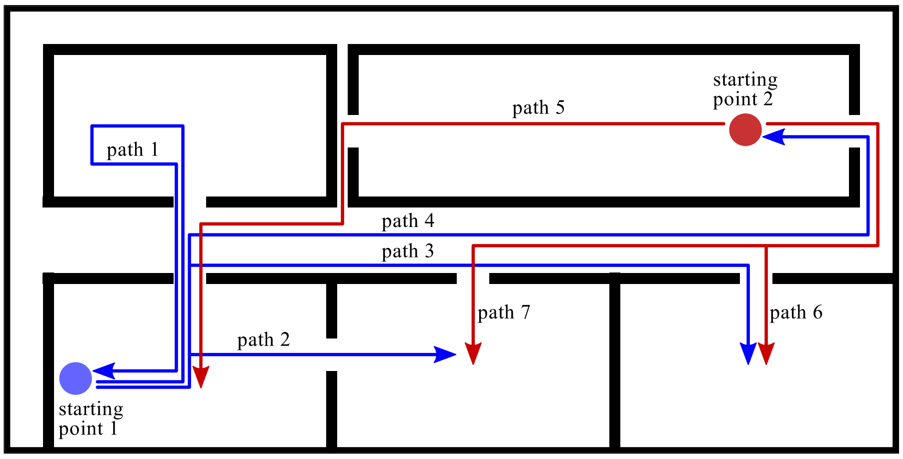


Figure 15: Common human trajectories used in the simulation environment. Note that path 1 in this figure corresponds to the example shown in Figure 12.

Evaluation metrics.

As a performance measure, we computed the reduction in path length to evaluate the efficiency of the generated motion actions, e.g., to assess battery saving. We compare the path resulting from applying our framework to the path generated from the direct-following strategy, in which the robot moves at each time step toward the node of the human's position. This presents the best heuristic behavior the robot can execute as the actual destination can only be predicted and might be wrong. We evaluated the statistical significance using a *two-tailed paired z-test*.

The experimental results show, as illustrated in Table 2, that our learned policy significantly outperforms the direct-following approach in terms of path length (statistical significance of 95%), which was reduced by 7.33%. Additionally, the percentage of runs in which the robot failed to reach the final destination of the user, is reduced to only 1% compared to 14% in the case of the direct-following approach.

Table 2: Path length reduction of our approach compared to directly following

Path reduction	**Statistical significance**	**Failed runs (our approach)**	**Failed runs (direct following)**
7.33%	95%	1%	14%

Heuristics.

Furthermore, we compare the performance of our approach to that of a heuristic that depends only on predicting the user's future location and moves the robot toward it. We evaluate this heuristic using the same test set and show the percentage of path reduction compared to direct following similar to the aforementioned evaluation of our approach. We consider two variants of this heuristic, one variant is that at every time step the robot moves one step toward the user's predicted location after few future time steps (three-step prediction ahead similar to the prediction used in the reinforcement learning approach, i.e., see Equation 30). The other variant is to continuously predict the final destination of the user and to move one step towards that predicted destination. We apply the prediction method illustrated in Section 5.3 to obtain both predictions.

Table 3: Path length reduction of a prediction heuristic compared to directly following

Prediction	**Path reduction**	**Statistical significance**	**Failed runs**
three-steps ahead	−15.03%	99%	4.45%
final destination	−35.56%	99%	5.05%

As illustrated in Table 3, the experimental results show that relying only on the prediction model yields a significantly worse performance even compared to the direct following technique since it suffers from prediction uncertainties. The first two columns of Table 3 show that both variants of the prediction heuristic approach lead to significantly longer paths (with a statistical significance of 99%) compared to the direct following due to the uncertainties involved in the predictions. Furthermore, the third column in Table 3 that the percentage of Unsuccessful runs is greater than that of our approach. Therefore, these results highlight the strength of our approach and its ability to deal with the prediction uncertainties to generate foresighted navigation actions that lead to efficient robot trajectories.

5.5.3 *Real-World Experiment*

In addition to the experiments in simulation, we performed an experiment with a real robot. We used a Robotino robot

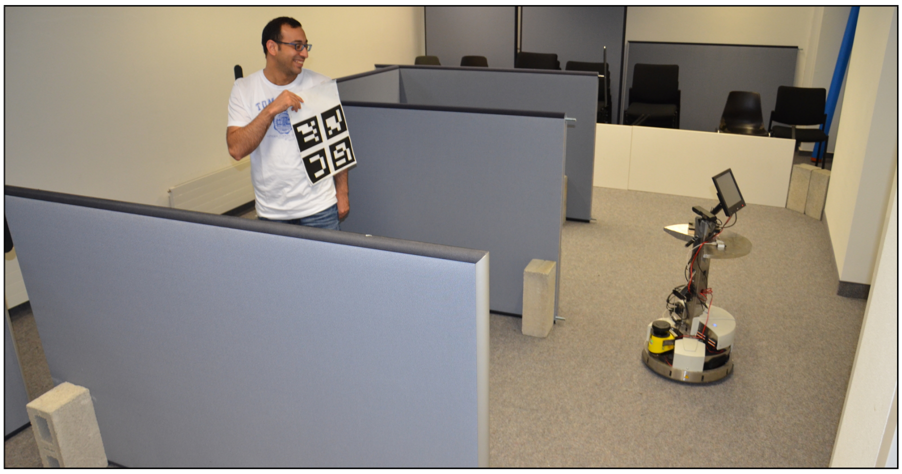


Figure 16: The robot and the user with a board of ArUco markers to simplify pose estimation in the real-world experiment.

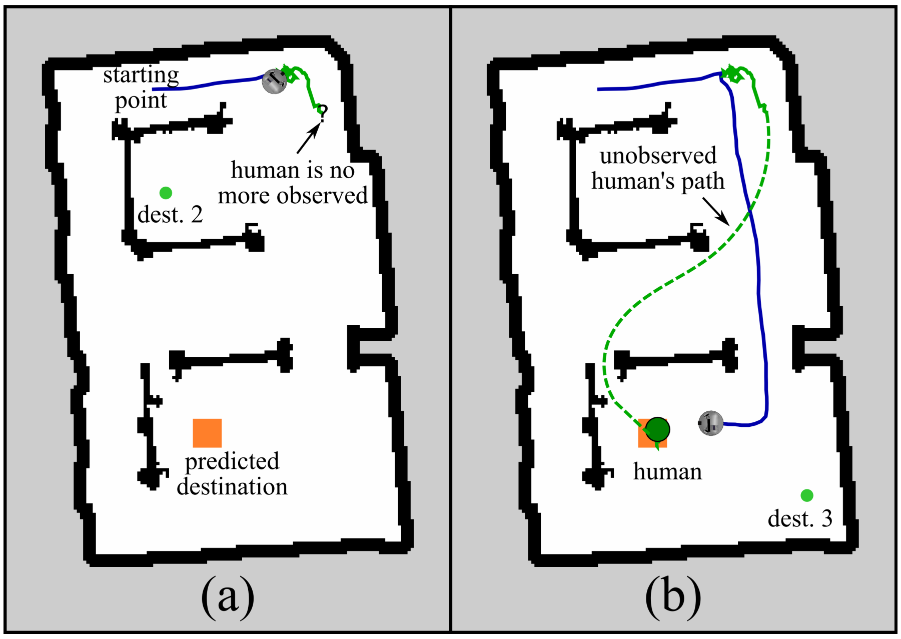


Figure 17: Real-world experiment: (a) The human gets outside the robot's camera field of view while walking along his path (green trajectory) toward his destination. The robot keeps updating the prediction until it reaches the correct destination. (b) The robot executes the learned actions to reach the destination successfully through an alternative path (blue trajectory) that avoids the narrow entrance that does not fit the robot size.

from Festo and used the on-board laser sensor for particle filter based localization on the given grid map. To estimate the human's position, the user holds a board of ArUco markers [Garrido-Jurado et al., 2014] as shown in Figure 16. We focused on showing the ability of our framework to handle situations in which the human's trajectory also contains passages that are impassable for the robot so that simply following the human at close distance will fail. The environment of the real-world experiment with the topo-metric graph is shown in the left image of Figure 14. The Q-table of the learned navigation actions for this environment was also learned in simulation.

For the experiment shown in Figure 17, the human moved toward his intended destination (which is unknown for the robot) through a path that involved passing through two narrow walls, which does not fit the size of the Robotino robot. Moreover, the human left the robot's field of view (due to the robot's orientation). The prediction, however, kept being updated until it reached a final destination. Meanwhile, the robot executed the learned Q-table actions based on the current robot position and the current prediction at each time step. As can be seen, the robot could successfully encounter the human at the intended destination. The robot took an efficient path that fits the robot size and at the same time considers potential prediction errors, i.e., it executed an observation action by rotating 360° that put destination 2 into its field of view (see Figure 17 (a)). After that, the learned actions led to a path that put destination 3 in the field of view of the robot as it moved on its path (see Figure 17 (b)). Therefore, if the prediction about the human pose was wrong and the human was waiting in any of these destinations, there would be a high probability of observing the human again and, thus, the prediction would have been updated accordingly. Note that in this scenario, if the robot navigates according to the strategy of closely following the human it would get stuck in front of the narrow passage.

5.6 CONCLUSIONS

In this chapter, we presented a reinforcement learning approach to generate efficient navigation actions for a service robot that needs to encounter a user at designated locations, between which a human moves on typical paths. We extended our previous work presented in Chapter 4 by using a more compact representation of the movement possibilities, modifying the state space, enhancing the reward function. Furthermore, we hereby apply a particle filter based prediction about the human's motion to deal with possible occlusions and generate effective navigation and observation actions.

In simulated as well as in real-world experiments, we showed that our method enables the robot to navigate efficiently to predicted destinations and also deal with cases in which the human is not within the robot's sensor range. In these situations, the robot explicitly considers to execute observation actions to re-observe the human and update the prediction about the destination. Additionally, the experimental results demonstrate that our framework generates significantly shorter paths and performs with a higher percentage of successful runs compared to a baseline approach in which the robot tries to follow the human at close distance. Furthermore, we compare the performance of our approach to that of a heuristic that depends only on predicting the user's future location and moves the robot toward it. We show in this comparison how such heuristic suffers from prediction uncertainties and on the other side how our approach is able to overcome such uncertainties and generate foresighted navigation actions that lead to efficient robot trajectories.

6 PEOPLE FINDING UNDER VISIBILITY CONSTRAINTS

In this chapter, we introduce our people finding approach which focuses on scenarios where the user moves along common paths between places where they remain for a while. Our approach simulates possible behaviors of the user for future time steps in addition to considering the visibility characteristics of the environment in order to select a good search location at which there is a high likelihood to observe the user.

6.1 INTRODUCTION

Finding a person is an essential functionality that is needed by several applications of mobile service robots. In scenarios where the robot aids the user with tasks that require the robot to move freely across the environment but also direct interaction at certain times, the robot needs a strategy to find the user as fast as possible when it is necessary. Typically, users do not stay at a fixed position but move along common paths between places where they remain for a while, e.g., to discuss work with a colleague, grab some material, or get a coffee. The robot needs a good strategy to find the user as fast as possible also in these situations to carry out its task. One possible solution to the search problem is to apply techniques that try to maximally cover the visible area of the environment [Choset, 2001; Guibas et al., 1996; Suzuki and Yamashita, 1992]. However, these approaches lead to long search times and high navigation costs as they aim at covering the whole environment. Moreover, the maximum coverage techniques will not necessarily revisit already covered regions and since the user is assumed to move freely across the environment, the robot might miss them during the search.

People finding is essential in daily tasks.

Existing approaches either focus on maximum coverage as mentioned above while ignoring making use of prior knowledge about frequently visited destinations of the user and their connecting paths, or they predict the user's most likely locations without considering the visibility constraints resulting from the environment layout as well as ignoring the time needed by the robot to reach such locations. Other approaches considered the frequency of human's presence at specific locations as well as the robot's field of view, however, they do not model the human's motion and therefore cannot predict their expected position at a certain intermediate time step. A detailed discussion about these approaches, which demonstrate the research gaps among them, can be found in Chapter 2.

Drawbacks of existing methods.

6.1.1 *Contributions*

In contrast to all such aforementioned methods, we make use of prior knowledge about frequently visited destinations of the user and their connecting paths to achieve a short search time. We developed an approach that determines good search locations using hidden Markov model (HMM) based predictions of the user's position on a graph representation of paths which the user typically takes. Our novel approach performs HMM-based simulations to compute the likelihood of the observability of the user at each possible location. We hereby take into account the time needed by the robot to reach the search locations from its current position as well as the visibility constrains that arise from the robot's limited field of view and obstacles.

6.1.2 *Framework Overview*

Figure 18 highlights the strength of our approach. The location of the user is initially unknown. The user may walk toward any of a set of predefined destinations, known by the robot. Thus, the exact intended destination and the user's location at each time step are unknown. The robot's task is to find the user as soon as possible. As illustrated, our approach leads to the selection of an effective search location that provides the highest expected observability, i.e., the robot can observe the corridor and the entrances to multiple rooms and, thus, locate the user.

Example of good search locations.

This search location selection can be achieved via performing HMM-based simulations, that rely on individual HMM-based predictions, in order to simulate possible behaviors of the user for future time steps. Our novel approach simulates the user's

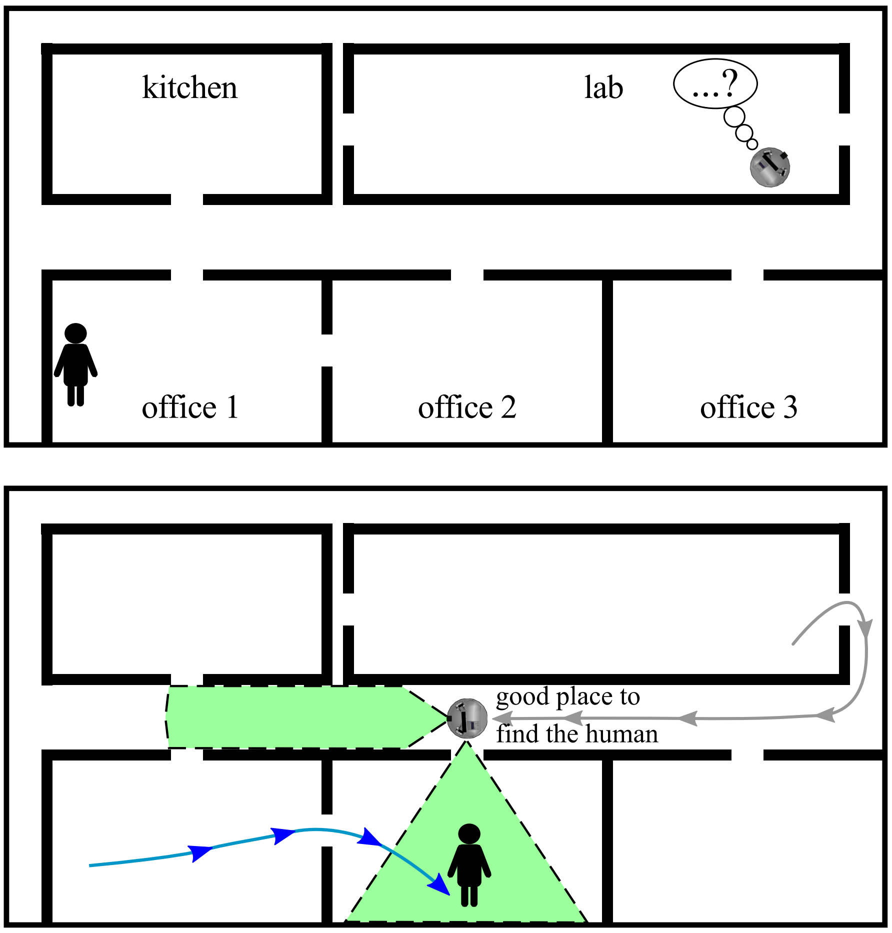


Figure 18: Top: The robot needs to find a user whose current location is unknown. The user may walk toward any of a set of predefined destinations, known by the robot. Bottom: The robot needs to select a good search location that covers most of the expected paths of the user. Our approach selects a search location with maximum observability of the user at the time the robot reaches it.

motion for future time steps and computes the predicted beliefs about the user's presence at each possible location at each future time step. Then, we make use of these predicted beliefs, i.e., that are obtained via simulations, while considering the visibility characteristics of the environment in order to select a good search location at which there is a high likelihood to observe the user. Moreover, our approach takes into account the time needed by the robot to reach a certain search location, from the robot's current location, which accordingly plays a key role in the selection criteria of that search location, i.e., reaching a search location either early or late leads to negative results even

HMM-based Simulations.

if that search location may be considered as a good choice from the prospective of the visibility characteristics of that location.

Concerning the individual predictions that are generated and accordingly used by our simulations, they are generated via a hidden Markov model (HMM) as mentioned above. Each hidden state of our proposed HMM represent a potential location of the user, i.e., each hidden state corresponds to one of the topo-metric graph nodes that represent the environment (see Figure 19). Therefore, the belief of being at any hidden state represents our belief about the user's presence at the corresponding location. Furthermore, the transition probabilities between these states model the user's motion along their path and, thus, these transition probabilities correspond to our prior knowledge about the user's frequently visited destinations and the typical paths between them. Accordingly, we can keep a multimodal belief about the user's presence following the beliefs of the hidden states, where these beliefs are updated at each time step based on the states' transitions, i.e., the transitions of our beliefs between the states model the user's motion at each time step. Thus, we can predict the user's future positions via performing the transitions of the beliefs between the hidden states for some future time steps and compute the resulting future beliefs of the user's presence.

HMM.

We show in extensive simulated experiments and in various environments that our technique generates search locations that significantly reduce the time to find the user compared to a greedy solution that is provided with background information about the possible destinations between which the user moves. In the experiments, we model noisy observations and dynamic obstacles to show the robustness of our approach.

6.1.3 *Organization*

The remainder of the chapter is structured as follows. The next section states the problem formulation in detail. Then, Section 6.3 discusses our graph-based people tracking model. After that, we introduce our novel approach to select effective search locations in Section 6.4. Then, we discuss the experimental setup and present comparative results in Section 6.5. Finally, we draw some conclusions in Section 6.6.

6.2 PROBLEM FORMULATION

The task of the robot is to find a non-stationary user as fast as possible. The environment is hereby known to the robot and it has prior knowledge about locations where the user frequently stays and their typical paths between these locations. We refer to these locations as *destinations*. After reaching such a destination, the user might stay there or move to another destination after a certain waiting time.

Frequently visited locations.

We represent the environment as a grid map with an overlaid topo-metric graph as shown in Figure 19, where each cell in that grid is mapped onto its closest graph node [Bayoumi et al., 2017]. The connections shown between neighboring nodes correspond to valid paths between these nodes. However, some of the paths are only passable by humans, e.g., due to the size of the robot or any other potential constraints of the searching environment.

Topo-metric graph.

The location of the user is initially unknown to the robot as well as their intended destinations when moving. After reaching a destination, the user might stay there or start moving to another destination after some time. We assume the moving velocity of the user to be within a certain range, however, the exact velocity of the user is unknown to the robot. Dynamic obstacles, e.g., other humans, can appear in the environment and temporarily constrain the robot's field of view. The task is considered as successful when the robot observes the user within its field of view.

Assumptions.

6.3 GRAPH-BASED PEOPLE TRACKING

To represent the belief about the location of the user and track its motion on the graph between the destinations, we apply a hidden Markov model (HMM). A hidden Markov model, in general, consists of a set of hidden states. These states are considered hidden since the current state of the model is unknown. Therefore, we estimate the current state based on subsequent observations. Furthermore, the model's state changes at each time step, i.e., which is called a state transition, based on a transition probability distribution. Thus, given an initial distribution that describes our belief about the initial hidden state of the model, we keep updating these beliefs while considering

HMM.

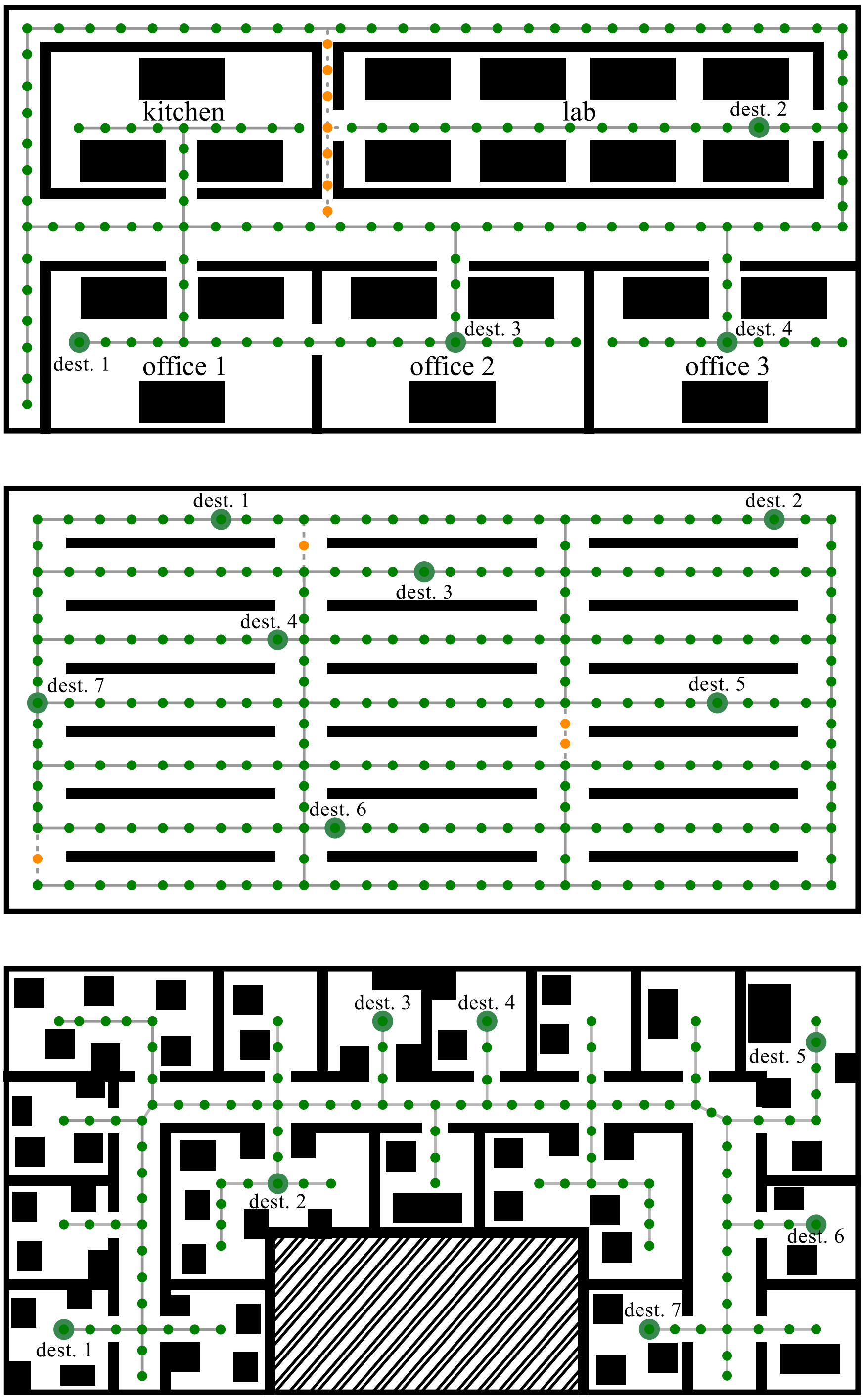


Figure 19: Three simulation environments with overlaid topo-metric graphs. Each environment is represented as a grid map with an overlaid graph, where each grid cell is mapped to the closest graph node (green dots) in the same room. The orange dots represent paths that are only passable by the user but not by the robot. The bold green dots represent the predefined destinations between which the user moves.

the possible transitions from each state to the other states and the likelihood of such transitions, as well as taking into account the likelihood of the generation of the obtained observations from each of these states [Rabiner and Juang, 1986].

We hereby apply a HMM instead of a particle filter to achieve more robust performance that is less prone to the initially generated particles and their propagation along the possible paths. Since an infinite number of particles is theoretically needed to fully represent the probability distribution of the user's presence at each possible location at each time step. Instead, this issue is implicitly handled in the case of HMMs.

In our proposed hidden Markov model, each hidden state corresponds to one of the topo-metric graph nodes, that represent the environment, in order to track the user's potential locations over that graph. Every node is represented by a unique hidden state. Accordingly, the belief of the occurrence of each hidden state represents our belief about the user's presence near to the corresponding graph node. Furthermore, we use the information about the typical paths between the destinations and the times that the user stays at the destinations to find the average time that the user occupies each node. Then, we generate the initial state distribution, i.e., which represents our belief about the user's unknown initial location, according to this occupation likelihood. For each node, we learn the state transition probability distribution which models the transition of the user from that node to the neighboring nodes based on the typical paths that lead through that node. Moreover, this transition probability distribution takes into account the average velocity of the user as well as the typical time the user spends at the frequently visited destinations.

Learning distributions.

At each time step, we update the belief based on the transition probabilities and the belief at the previous time step as follows:

States' transitions.

$$Bel_t(i) = \sum_j p(i \mid j)\ Bel_{t-1}(j), \quad \forall i \in \mathcal{N}, \tag{32}$$

where $\mathrm{Bel}_t(i)$ is the belief of being at node i at the time step t and $p(i \mid j)$ is the transition probability from node j to node i. Additionally, $\mathcal{N}$ is the set of graph nodes.

After that, we update the belief of the graph nodes proportional to the observation likelihood. According to our problem, positive observation of the user leads to a successful termina-

Negative observations.

tion of the search process. Therefore, we only consider negative observations in the observation model to update the belief. For the graph nodes that fall within the robot's field of view while the user is not currently detected, the probability is reduced, as follows:

$$Bel_t(i) = \begin{cases} \gamma Bel_t(i), & \text{if } (\mathrm{i} \in \mathcal{FOV}) \wedge \textit{user not detected} \\ Bel_t(i) & \text{otherwise} \end{cases}, \qquad (33)$$

False negatives. where $\mathcal{FOV}$ is the area covered by the robot's visual sensors and $\gamma \in [0, 1)$ is a reduction factor. Since the likelihood of false negative observations increases with the distance of the user to the robot, γ decreases with this distance. As we assume a proper identification system, we do not model false positive ob- *False positives.* servations. Note, however, that we can deal with false positive observations for a short time by requiring a minimum number of subsequent time steps where the human is detected before the search is assumed to be successful.

Then, we can estimate the most likely location of the user by considering the node with the highest probability. However, moving the robot to that estimated location is inefficient since this strategy ignores the time needed to reach the search loca- *Most Likely Location.* tion and to find the user. Moreover, estimating the most likely location of the user at each time step and accordingly updating the robot's search location at each time step might lead to an oscillating navigation behavior as the estimation might jump across the map. Therefore, we present in the next section our approach that performs HMM-based simulations using the individual predictions obtained from the HMM.

6.4 SELECTING SEARCH LOCATIONS

In this section, we describe our approach to selecting effective search locations for the robot to find the human. Relying only on the estimated most likely location of the user at each time step leads to an oscillating navigation behavior as the estimation might jump across the map, i.e., following the approach of Goldhoorn et al. [2017] (refer to Section 2.2 for more details). We, therefore, propose in Section 6.4.1 a method that performs HMM-based simulations which take into account the time needed by the robot to reach the possible search locations from its current place. After that, we present in Section 6.4.2

few possible extensions to that proposed approach to enhance its performance.

6.4.1 *Using HMM-based Simulations*

We first perform HMM-based simulations to compute the likelihood of the user's presence for each graph node at future time steps. Thus, we use the HMM presented in the previous section in order to compute this likelihood. We predict the human's motion along the graph by simulating the state transitions for future time steps and compute the probability of the graph nodes according to Equation 32. The belief at any time step represents the estimate about the user's presence at the corresponding graph node at that time step. We simulate the states transitions as many future time steps as needed by the robot to reach the furthest graph node relative to the robot's current node.

HMM-based Simulations.

After that, we use the predicted beliefs in order to compute the likelihood of observing the user from each graph node, i.e., the likelihood of the user's observability at that node, while considering the time needed to reach this node. For example, if a node lies ten time steps away, we consider the computed belief resulting from simulating the state transitions ten time steps into the future when computing the likelihood of the user's observability at this node.

Our approval would be to compute the observability likelihood l_k of the user at each node k as follows:

Observability likelihood.

$$l_k = \sum_{i \in \mathcal{O}_k^{\mathcal{T}}} Bel_{t+\mathcal{T}}(i), \quad \forall k \in \mathcal{R}^{\mathcal{T}}, \quad 1 \leqslant \mathcal{T} \leqslant T, \tag{34}$$

where $\mathcal{R}^{\mathcal{T}}$ is the set of graph nodes that can be reached from the robot's current node n_r within exactly $\mathcal{T}$ future time steps, T is the number of future time steps needed by the robot to reach the furthest graph node from n_r, and $\mathcal{O}_k^{\mathcal{T}}$ is the group of graph nodes that can be observed from node k by performing an observation action, i.e., a full rotation. Moreover, $Bel_{t+\mathcal{T}}(i)$ is the predicted belief about the user's presence at node i after $\mathcal{T}$ future time steps relative to the current time step t.

After computing l_k for every k, we select the graph node with the highest observability likelihood s as the next search location[1], i.e.,

Search location selection.

$$s = \operatorname*{argmax}_k l_k. \tag{35}$$

The pseudo-code of our search goal selection algorithm is listed in Algorithm 2. As can be inferred from the example shown in Figure 20, the robot selects the next search goal as the location that is expected to provide highest observability at the time the robot reaches it.

Algorithm 2 : Selection of the next search location using HMM-based simulations

```
Input  : beliefs and robotPose
Output : next search location
likelihood ← {};
for t ← 1 to T do
    beliefs ← simulate HMM's states transitions for one
      more time step in the future;
    reachableNodes ← nodes that can be reached by the
      robot in exactly t time steps;
    // calculate observability likelihood for each node;
    for r ∈ reachableNodes do
        visibleNodes ← visible nodes from r (incl. r);
        for v ∈ visibleNodes do
            likelihood[r] ←
              likelihood[r] + beliefs[v];
        end
    end
end
return $\operatorname*{argmax}_{node}$ likelihood[node];
```

The robot then navigates to the selected node along the shortest path in the graph and does an observation action by performing a full rotation. If the user cannot be found anywhere on the way to the current search location nor while performing the observation action, a new search location is selected as previously and so forth. Performing the HMM-based simulations in the described way provides an effective method for select-

1 Note that the time is inherently considered in the computation of the l_k, such that s does not need to have a time index.

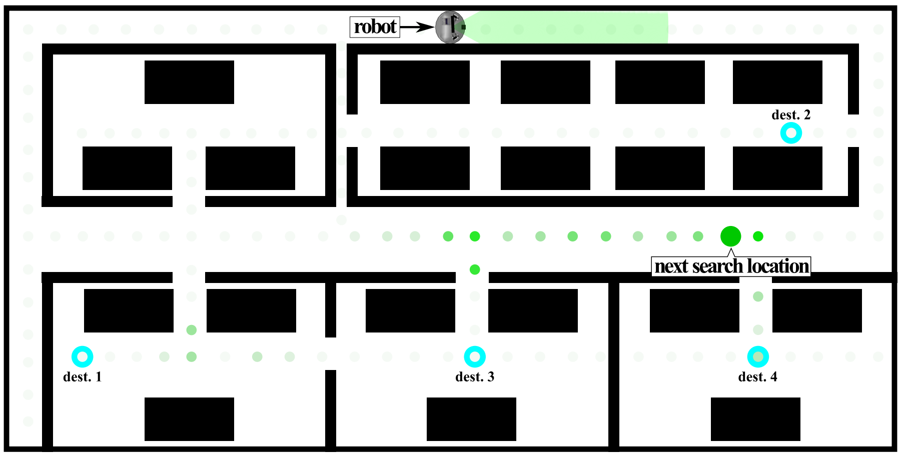


Figure 20: This figure shows the selected search location according to Algorithm 2. The graph nodes are drawn with a color intensity corresponding to the observability likelihood of the user at the time the robot reaches this search location. The robot selects the node that provides the highest observability likelihood as next search location.

ing a good search location that takes into account the dynamic behavior of the user.

While computing the next search location, we do not consider waiting actions or non-shortest paths, as this results in infinite possibilities to reach any node. Neither do we take into account the observability along the intermediate nodes to the considered search location. As we have found out in our experiments, this leads to a selection of search locations with longer paths and does not decrease the search time.

6.4.2 *Extended Version of HMM-based Simulations*

Moreover, as an extended version of our aforementioned approach, we extend the computations of the user's observability likelihood as opposed to Equation 34 such that we additionally consider the user's observability likelihood at each of the intermediate nodes along the robot's path to the search location instead of only considering it at the search location itself. However, this leads the robot to favor far search locations despite of the deficiency in order to maximize the observability likelihood accumulated along the path, as we have mentioned previously. Therefore, in order to overcome such issue, we consider the av-

Observability along the robot's path.

erage of the HMM's predicted beliefs of every observed node along the robot's path, i.e., which represent the likelihood of the user's presence there, at the corresponding time steps of being observed or visited by the robot. Additionally, we take into account the impact of observing the same nodes for multiple times along the path of the robot via applying a reduction factor to alleviate such effect. These repetitions cause a problem since the predicted belief of a node, that is observed repeatedly for multiple times, will be included in the aforementioned averaging step for multiple times as well which, accordingly, increases the uncertainty involved in the search location selection. Therefore, this reduction factor, which is computed independently for each path, corresponds to the ratio between the number of the observed intermediate nodes without repetition to that with repetition. For example, this ratio takes the value of 1 only if each observed intermediate node is only observed once during the robot path to its search location. Therefore, we compute the likelihood of observability, as opposed to Equation 34, as follows:

$$l_k = \sum_{i \in \mathcal{O}_k^{\mathcal{T}}} Bel_{t+\mathcal{T}}(i) + \frac{\alpha}{\mathcal{T}-1} \sum_{\delta=1}^{\mathcal{T}-1} \sum_{i \in \mathcal{Q}_k^{t+\delta}} Bel_{t+\delta}(i), \qquad \forall k \in \mathcal{R}^{\mathcal{T}},\ 1 \leqslant \mathcal{T} \leqslant T, \tag{36}$$

where

$$\alpha = \frac{\sum_{\delta=1}^{\mathcal{T}-1} \sum_{i \in \mathcal{Q}_k^{t+\delta}} \beta_i}{\sum_{\delta=1}^{\mathcal{T}-1} |\mathcal{Q}_k^{t+\delta}|}, \tag{37}$$

and

$$\beta_i = \begin{cases} 1 & \text{if first time to observe } i \\ 0 & \text{otherwise} \end{cases}. \tag{38}$$

Again, $\mathcal{R}^{\mathcal{T}}$ is the set of graph nodes that can be reached from the robot's current node n_r within exactly $\mathcal{T}$ future time steps, and T is the number of future time steps needed to reach the furthest graph node from n_r. Moreover, $\mathcal{O}_k^{\mathcal{T}}$ is the group of graph nodes that can be observed from node k by performing an observation action. On the other hand, $\mathcal{Q}_k^{t+\delta}$ is the list of nodes that are observed by the robot at the intermediate time step $t+\delta$ along its path from its current graph node n_r to node k. Additionally, $Bel_{t+\mathcal{T}}(i)$ is the HMM's predicted belief about the user's presence at node i after $\mathcal{T}$ future time steps relative to the current time step t. The reduction factor α is computed independently for each possible path. After that, the robot selects its next search location that maximizes the observability likelihood following Equation 35.

Intermediate observation actions.

Additionally, we consider the possibility of performing observation actions, i.e., a full rotation, at important locations. Therefore, during the HMM-based simulations we consider the intersection nodes at which performing an observation action might lead to a considerable information gain. We select intersection nodes eligible for observation actions during the HMM-based simulations based on the predicted beliefs of the user's presence at the graph nodes that can be observed only if an observation action is performed by the robot as follows:

$$\mathcal{Q}_k^{t+\delta} \leftarrow \begin{cases} \mathcal{O}_k^{t+\delta} & \text{if } \sum_{i\in\mathcal{O}_k^{t+\delta}} Bel_{t+\delta}(i) > c \sum_{i\in\mathcal{F}_k^{t+\delta}} Bel_{t+\delta}(i) \\ \mathcal{F}_k^{t+\delta} & \text{otherwise} \end{cases}, \quad (39)$$

where, $\mathcal{F}_k^{t+\delta}$ is the set of nodes that can be observed without a rotation needing to be executed by the robot at the intermediate time step $t+\delta$ along the robot's path from current graph node to node k. Furthermore, c is a constant threshold that controls the possibility of executing the rotation.

In other words, if the beliefs of such nodes, i.e., that can be only observed via a rotation, are greater than a certain threshold that involves the beliefs of the nodes that are already observed without such rotation, then an observation action is considered. Accordingly, the list of intersection nodes along the robot's path at which observation actions are to be performed is determined for each reachable node, i.e., while computing the observability likelihood of the user at each of them. Then, the actual list to be executed is chosen corresponding to the selected search location.

It is worth mentioning that we do not consider updating the search location before it is reached by the robot even when the likelihood of observability of that location loses considerable amount of its power. As we have found in our experiments, aborting the search location and selecting a new one has a negative impact on the search time and this impact appears significant in environments where there are numerous movement probabilities as in the case of the second environment (see Figure 19). These aborting decisions are considered as hurried decisions that ignore the simulations based on which the search locations are selected. Furthermore, these hurried decisions will bias the subsequent selection of search locations since the same location will not be reselected even if it is a good one because of the effect of the negative observations on our belief of the so far observed intermediate nodes.

In the next section, we present an extensive evaluation of both variants of our approach to selecting the next search goal and compare them to a greedy approach with background information and the approach of Goldhoorn et al. [2017] that relies on the estimation of the user's current location without performing any simulations.

6.5 EXPERIMENTAL RESULTS

We carried out extensive experiments to evaluate each of our proposed approaches and compare them to alternative methods in different environments. In Section 6.5.1, we demonstrate the setup of our simulation experiments. Then, we discuss the achieved results in Section 6.5.2.

6.5.1 *Experimental Setup*

Simulation environments.

We performed the experiments in three different, challenging simulation environments (see Figure 19), each of size $41\,\text{m} \times 20.5\,\text{m}$ with a grid map resolution of $0.25\,\text{m}$ and a node distance of $1.5\,\text{m}$. In the first two environments, multiple paths exist between the destinations, among which the user chooses one based on a certain known probability distribution. The transition probabilities of the user from one destination to the others are equally likely. Note, however, that some passages are impassible to the robot, i.e., the dotted line with orange nodes for the first two environments to make the search problem even more challenging. As can be seen, the visibility characteristics of the second environment are very difficult due to the cluttered nature of the environment's walls. On the other hand, the third environment is much simpler to navigate as there is only one path between any two nodes for both the robot and the user.

In each experiment, the position of the user is initialized according to the occupation likelihood (see Section 6.3) and the user moves in the environment between the predefined destinations. The user does not necessarily move on the shortest path but might take detours. When the user reaches their destination, they wait there for a certain period of time. The user repeats this behavior until they reach their fourth destination and remains there. The velocity of the user is sampled from a certain interval. At each time step, the position of the user is mapped onto the closest graph node given its grid map position. The initial location of the user is unknown to the robot and is outside its field of view.

Occupation likelihood.

The beliefs of the hidden states of the HMM are initialized and updated as described in Section 6.3. The number of dynamic obstacles that constrain the robot's field of view ranges from three to five and their velocities are sampled from the same interval as the velocity of the user.

Dynamic obstacles.

Furthermore, we perform observation actions at the intermediate intersection nodes if the sum of beliefs of the hidden states corresponding to the nodes that can be observed from such rotation is at least twice those which can be observed without rotation, i.e., constant c in Equation 39.

The search task is considered successful when the robot observes the user. The robot's field of view has a horizontal opening angle of $58°$, which corresponds to that of an *ASUS Xtion Pro Live* RGB-D sensor, and a $10\,\text{m}$ view distance. We set the probability of false negatives between 0.05 and 0.15 linearly increasing with the distance between the robot and the user. We do not consider false positive observations in the simulation experiments. However, we can deal with false positive observations in real-world scenarios for a short time by requiring a minimum number of subsequent time steps where the human is detected before the search is assumed to be successful.

6.5.2 *Results and Evaluation*

We performed $5,000$ experiments in each of the three environments. In order to evaluate the performance of both versions of our approach, we considered the search time of each of them and compared it to the time needed by two different approaches. The strategy of the first alternative approach is to visit all destinations in a greedy fashion using background information, i.e., the knowledge about the destinations of the user. The

Benchmarks.

greedy approach does not consider any prediction about the user's location; it keeps selecting the closest unvisited destination as a search location until the user is found. After visiting all destinations, it starts the search process all over again.

We additionally compared our approach to a method that uses the hidden Markov model representation to infer the currently most likely location of the user and moves the robot toward that location for one time step, then updates the estimation and so on. This method is similar to the approach of Goldhoorn et al. [2017] and we refer to this method as the *one-step to estimation* method.

We hereby refer to the initial version of our approach that is presented in Section 6.4.1 as the *limited version*. On the other side, we refer to the other version which is discussed in Section 6.4.2 as *full version*.

We evaluated the statistical significance of our comparative experiments with a *two-tailed paired t-test*. The experimental results show that both variants of our method performing HMM-based simulations significantly outperforms each of the other two approaches with a statistical significance of 99%. Figure 21 shows the average relative search times achieved by both versions of our approach for each of the three environments with respect to the greedy approach with background information. Additionally, Figure 22 shows the average relative search times with respect to the one-step to estimation method. The results demonstrate that both variants of our approach, performing HMM-based simulations and predicting the observability likelihood of the user, significantly outperform all the other methods in various scenarios and environments regardless of their visibility characteristics.

Moreover, the extended version of our HMM-based simulations approach performs significantly better in all environments compared to the initial version with a statistical significance of 99% as well. These results show that making use of intermediate nodes, i.e., to gain more information via observation actions, is more efficient regardless of the environment type and its visibility characteristics.

As both versions of our approach do not guarantee to cover the entire map and, thus, might not find a search location close to the user's final destination, we switch to the greedy approach after a given time limit. The maximum time limit was determined experimentally such as to minimize the overall search

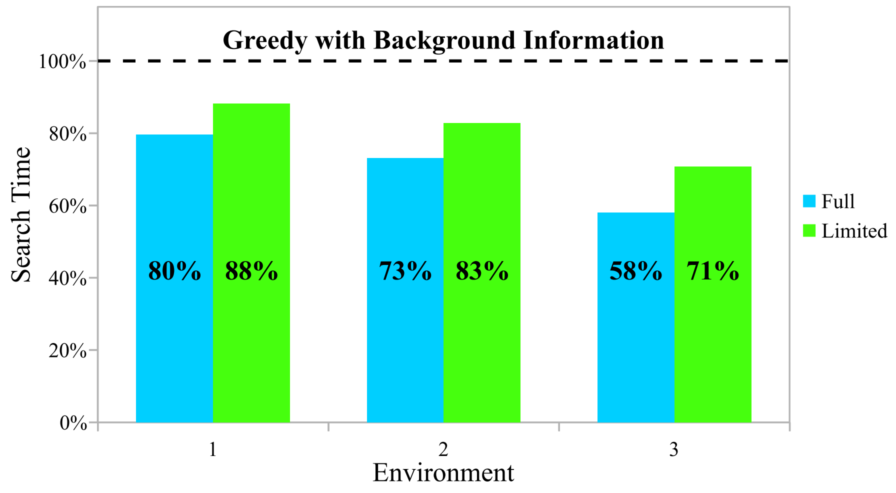


Figure 21: Average relative search time achieved by both versions of our HMM-based simulations approach with respect to the greedy approach with background information. The times are normalized so that the greedy approach equals 100%.

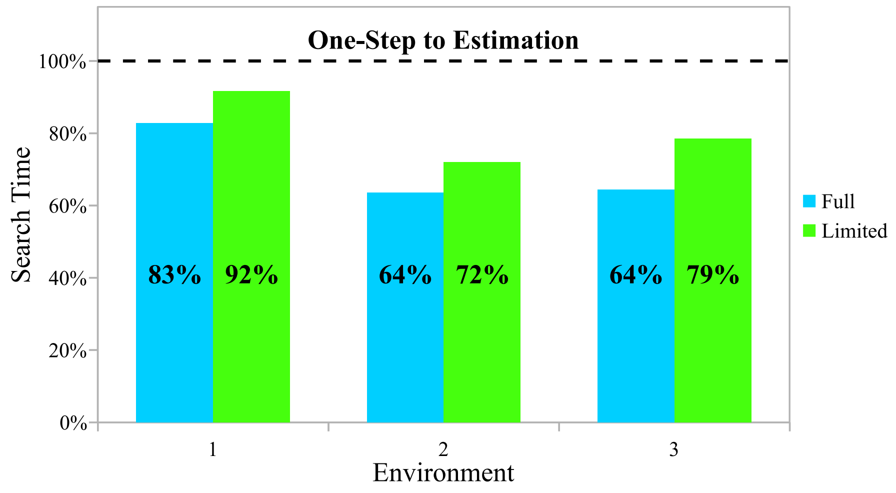


Figure 22: Average relative search time achieved by both versions of our HMM-based simulations approach with respect to the one-step to estimation method. The times are normalized so that the one-step to estimation approach equals 100%.

time. Table 4 shows the percentage of experimental runs using our approaches and the "one-step to estimation" approach that exceeded this time limit and switched to the greedy method. As shown, our techniques based on HMM simulations outperform the one-step to estimation method for all the environments. Fur-

Table 4: Percentage of switching to the greedy approach.

	HMM-based Simulations		**One-Step to Estimation**
	Full	**Limited**	
Env. 1	0.8%	1.77%	4.38%
Env. 2	1.47%	1.75%	4.88%
Env. 3	4.2%	8.25%	12.15%

thermore, the extended version of our approach outperforms the initial version for all the environments as well. It is worth mentioning that we included these additional search times resulting from switching to the greedy approach in all of the previously presented results.

A video showing the advantages of our approach for a comparative example run can be downloaded from `https://www.hrl.uni-bonn.de/ias18bayoumi.mp4`. It demonstrates the superior performance of our novel search method.

6.6 CONCLUSIONS

In this chapter, we presented an approach that enables a mobile robot to quickly find a non-stationary user in complex environments. Our approach selects the best next search location by predicting future paths of the user. To compute the likelihood of the observability of the user at possible search locations, we apply HMM-based simulations using hidden Markov model (HMM) on a graph representation of possible paths in the environment. We hereby take into account the time needed by the robot to reach the search locations as well as visibility constraints. Furthermore, we present an extended version of our HMM-based simulations approach that takes into account the user's observability likelihood at each of the intermediate nodes along the robot's path. Additionally, we consider performing observation actions at important locations, i.e., intersection graph nodes, along the robot's path to its search location in order to gain more information about the user's location.

As our simulation experiments demonstrate, our approach enables the robot to select effective search locations to find the user within a short amount of time. We showed in extensive experiments that both versions of our proposed method significantly outperforms two other common search methods, i.e., a greedy approach using background information and a method that infers the currently most likely location of the user and moves the robot toward that location for one time step. The experiments included runs where occlusions caused by dynamic obstacles as well as false negative detection occurred, which will be the case for real-world scenarios.

7 CONCLUSIONS & OUTLOOK

In this dissertation, we presented people following and finding approaches that greatly reduce the robot's navigation costs and search time, respectively. We considered each of the following and finding tasks separately. Concerning people following, as opposed to existing approaches, we presented the first framework, to the best of our knowledge, that focuses on the efficiency of the robot's trajectory during the following task. Thus, we presented a reinforcement learning approach to generate foresighted navigation actions. We considered the scenario in which the user moves between different designated locations where he might need to interact with the robot. Thus, the task of the robot is to meet the user at his intended destination which is initially unknown to the robot. Our approach constantly predicts the user's location after some future time steps, and applies reinforcement learning on the top of such prediction to generate foresighted navigation actions that are robust to the prediction uncertainties as well as the unexpected behaviors and detours of the user.

Furthermore, we addressed the people following problem within two main milestones. In the first milestone of developing such approach, due to the complexity of the problem and our focus on exploring the feasibility of relying on such learning approach, we assumed having perfect knowledge about the poses of both the user and the robot. In the second milestone, we relied on the robot's on-board sensors for active localization of the user and thus we considered dealing with occlusions and impassable paths to the robot, i.e., due to size or safety constraints. Additionally, we considered a compact representation of the movement possibilities via a grid map with an overlaid topo-metric graph, such that each grid cell corresponds to the nearest graph node. Such representation allows us to handle large and complex environments without overloading the learning process, as opposed to relying only on grid maps as in the first milestone. Furthermore, we applied a particle filter based prediction about the user's motion to deal with possible occlusions and generate effective navigation and observation actions. Such prediction model maintains a multi-modal belief about the user's location and it able to correct itself based on observation actions. Thus, particle filter based prediction is more robust compared to the softened Markov decision process prediction model that is used in the first milestone.

In simulated as well as in real-world experiments, we showed that our method enables the robot to navigate foresightedly leading to significantly shorter paths. Additionally, our method performed well in handling occlusions due to obstacles, besides, finding alternative paths to impassable paths through which the robot is not allowed to drive and follow the user. Our method showed that it could autonomously select key locations (e.g., intersection points, corridors ... etc) to perform observation actions to gain more information, i.e., either positive or negative information, to improve its belief about the user's location. Moreover, as our experiments showed, the generated actions could balance smartly between contradicting objectives. They tended to keep the user within the robot's limited field of view as much as possible while minimizing the navigation costs needed to meet the user at his intended destination. Moreover, we showed that our approach significantly outperforms a heuristic technique that depends solely on the predictions of the user's future location, i.e., performs path planning based on these predictions. This shows that our approach is able to overcome prediction uncertainties and generate foresighted navigation actions that lead to efficient robot trajectories.

On the other side, we presented an approach for people finding that significantly reduces the search time to find a moving user. Our approach relies on performing hidden Markov model (HMM) based predictions in order to select good search locations from which the user can be most likely observed. We assumed having prior knowledge about the user's common paths in the environment. Accordingly, we performed HMM-based simulations on a graph representation of possible paths in the environment using HMM-based predictions about the user's future locations. Additionally, we took into account the time needed by the robot to reach the possible search locations as well as the underlying visibility constraints of the environment. Moreover, we extended this proposed approach by considering the user's observability likelihood at each of the intermediate nodes along the robot's path to its selected search location. Furthermore, in the extended version, we allowed performing observation actions at visually important locations along the robot's path to the search location, as opposed to performing them only at the search location in the initially presented version of this approach.

Additionally, we performed extensive experiments to compare both versions of our approach to a greedy approach that is provided with background information about the possible destinations between which the user moves. This greedy approach does not perform any prediction about the user's location to generate the search actions. Additionally, we compared them to heuristics that perform path planning relying solely on the most likely location of the user, i.e., without performing HMM-based simulations. Both versions of our approach significantly outperformed all such techniques. Furthermore, we showed that the extended version of our approach significantly outperforms its initial version.

OUTLOOK

We hope that this dissertation could contribute to the scientific society and that other researchers will find this work helpful for their applications as well as a valuable starting point for a new research direction. There is still much to be done in this field and it may move toward many different directions, thus, in the rest of this section we point out some possible research directions that make use of our presented work.

In this dissertation each problem is discussed independently following a divide and conquer strategy. A possible starting point is to combine methods of people following and finding and address other domains of applications and problems. One possible example is to integrate our people following and people finding frameworks (see Chapter 5 and Chapter 6, respectively) into one system that can automatically switch between each of them according to the intended application of the robot. This will require designing the knowledge transfer between these two frameworks at each switch, e.g., the beliefs about the user's existence during a search task can be improved if it starts with an initial distribution that takes into account the user's last location that was observed during a previous following task.

One more interesting research direction is to consider applying deep reinforcement learning techniques, i.e, following the current upward trend of deploying deep learning techniques, in order to investigate possible performance improvements. Additionally, it may be useful to increase the complexity of the following scenarios by considering dynamic obstacles and periodic variations to the given static maps.

One more starting point is to combine our presented approaches with simultaneous localization and mapping (SLAM) techniques. Such that they no longer depend on a given map that is subject to change. Such feature will be necessary in order to obtain a robot that can apply both of our frameworks to any possible scenario.

Additionally, more specific applications of our work can be addressed, for example, the applications that target helping the children suffering from autism. Such an application domain is expected to have an increasing attention in the future due to the remarkable success of using robots in helping those children more than humans [Kim et al., 2013]. One example application can be a hide-and-seek game that specially addresses the children with autism where the robot tends to enhance the engagement of the children via teaching them how to take part and play such game. Such application is expected to have a high impact on stimulating the children's ability to communicate effectively.

BIBLIOGRAPHY

Bayoumi, AbdElMoniem and Maren Bennewitz (2015). "Efficient Human Following Using Reinforcement Learning." In: *Proc. of the Machine Learning in Planning and Control of Robot Motion Workshop (MLCP) at the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*.

Bayoumi, AbdElMoniem and Maren Bennewitz (2016). "Learning optimal navigation actions for foresighted robot behavior during assistance tasks." In: *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pp. 207–212.

Bayoumi, AbdElMoniem, Philipp Karkowski, and Maren Bennewitz (2017). "Learning Foresighted People Following under Occlusions." In: *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 6319–6324.

Bayoumi, AbdElMoniem, Philipp Karkowski, and Maren Bennewitz (2018). "People Finding under Visibility Constraints using Graph-Based Motion Prediction." In: *Proc. of the Int. Conf. on Intelligent Autonomous Systems (IAS)*. accepted.

Bayoumi, AbdElMoniem, Philipp Karkowski, and Maren Bennewitz (submitted in 2018). "People Finding under Visibility Constraints using Hidden Markov Models." In: *Robotics & Autonomous Systems*.

Bennewitz, Maren, Wolfram Burgard, Grzegorz Cielniak, and Sebastian Thrun (2005). "Learning Motion Patterns of People for Compliant Robot Motion." In: *Int. Journal of Robotics Research (IJRR)* 24.1.

Best, Graeme and Robert Fitch (2015). "Bayesian intention inference for trajectory prediction with an unknown goal destination." In: *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 5817–5823.

Boukhers, Zeyd (2017). *3D Trajectory Extraction from 2D Videos for Human Activity Analysis*. Vol. 44. Logos Verlag Berlin GmbH.

Busoniu, Lucian, Robert Babuska, and Bart De Schutter (2008). "A comprehensive survey of multiagent reinforcement learning." In: *IEEE Transactions on Systems, Man, And Cybernetics-Part C: Applications and Reviews, 38 (2), 2008*.

Choi, Byoung-Suk, Joon-Woo Lee, and Ju-Jang Lee (2010). "Effective target-following schemes for indoor mobile robots."

In: *Proc. of the IEEE Int. Conf. on Information and Automation (ICIA)*, pp. 666–671.

Choset, Howie (2001). "Coverage for robotics – A survey of recent results." In: *Annals of Mathematics and Artificial Intelligence* 31.1, pp. 113–126. ISSN: 1573-7470.

Fentanes, Jaime, Bruno Lacerda, Tomáš Krajník, Nick Hawes, and Marc Hanheide (2015). "Now or later? Predicting and maximising success of navigation actions from long-term experience." In: *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pp. 1112–1117.

Garrido-Jurado, Sergio, Rafael Muñoz-Salinas, Francisco Madrid-Cuevas, and Manuel Marín-Jiménez (2014). "Automatic generation and detection of highly reliable fiducial markers under occlusion." In: *Pattern Recognition* 47.6, pp. 2280–2292. ISSN: 0031-3203.

Ghahramani, Zoubin (2004). "Unsupervised learning." In: *Advanced lectures on machine learning*. Springer, pp. 72–112.

Giannoccaro, Ilaria and Pierpaolo Pontrandolfo (2002). "Inventory management in supply chains: a reinforcement learning approach." In: *Int. Journal of Production Economics* 78.2, pp. 153–161.

Goldhoorn, Alex, Anaís Garrell, René Alquézar, and Alberto Sanfeliu (2014). "Continuous Real Time POMCP to find-and-follow People by a Humanoid Service Robot." In: *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*.

Goldhoorn, Alex, Anaís Garrell, René Alquézar, and Alberto Sanfeliu (2017). "Searching and tracking people in urban environments with static and dynamic obstacles." In: *Robotics & Autonomous Systems* 98, pp. 147–157.

Guibas, Leonidas, Jean Latombe, Steven LaValle, David Lin, and Rajeev Motwani (1996). "A visibility-based pursuit-evasion problem." In: *Int. Journal of Computational Geometry and Applications*, pp. 471–494.

Harmati, István and Krzysztof Skrzypczyk (2009). "Robot team coordination for target tracking using fuzzy logic controller in game theoretic framework." In: *Robotics and Autonomous Systems* 57.1, pp. 75–86.

Hollinger, Geoffrey, Sanjiv Singh, and Athanasios Kehagias (2010). "Improving the Efficiency of Clearing with Multi-agent Teams." In: *Int. Journal of Robotics Research (IJRR)* 29.8, pp. 1088–1105.

Hollinger, Geoffrey, Srinivas Yerramalli, Sanjiv Singh, Urbashi Mitra, and Gaurav Sukhatme (2015). "Distributed Data Fusion for Multirobot Search." In: *IEEE Transactions on Robotics* 31.1, pp. 55–66. ISSN: 1552-3098.

Hornung, Armin, Maren Bennewitz, and Hauke Strasdat (2010). "Efficient Vision-based Navigation – Learning about the Influence of Motion Blur." In: *Autonomous Robots* 29.2, pp. 137–149.

Huang, Jiangyang, Shane Farritor, Ala' Qadi, and Steve Goddard (2006). "Localization and follow-the-leader control of a heterogeneous group of mobile robots." In: *IEEE/ASME Trans. on Mechatronics* 11.2.

Huang, Loulin (2009). "Control approach for tracking a moving target by a wheeled mobile robot with limited velocities." In: *Control Theory & Applications, IET* 3.12, pp. 1565–1577.

Kaelbling, Leslie, Michael Littman, and Andrew Moore (1996). "Reinforcement learning: A survey." In: *Journal of artificial intelligence research* 4, pp. 237–285.

Kim, Elizabeth, Lauren Berkovits, Emily Bernier, Dan Leyzberg, Frederick Shic, Rhea Paul, and Brian Scassellati (2013). "Social robots as embedded reinforcers of social behavior in children with autism." In: *Journal of autism and developmental disorders* 43.5, pp. 1038–1049.

Kolling, Andreas and Stefano Carpin (2008). "Multi-robot surveillance: An improved algorithm for the GRAPH-CLEAR problem." In: *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pp. 2360–2365.

Krajník, Tomáš, Miroslav Kulich, Lenka Mudrová, Rares Ambrus, and Tom Duckett (2015). "Where's waldo at time t? Using spatio-temporal models for mobile robot search." In: *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pp. 2140–2146.

Kretzschmar, Henrik, Markus Kuderer, and Wolfram Burgard (2014). "Learning to predict trajectories of cooperatively navigating agents." In: *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*. IEEE, pp. 4015–4020.

Kuderer, Markus and Wolfram Burgard (2014). "An Approach to Socially Compliant Leader Following for Mobile Robots." In: *Social Robotics*. Ed. by Michael Beetz, Benjamin Johnston, and Mary-Anne Williams. Vol. 8755. Lecture Notes in Computer Science. Springer International Publishing.

Kulich, Miroslav, Tomáš Krajník, Libor Přeučil, and Tom Duckett (2016). "To Explore or to Exploit? Learning Humans' Behaviour to Maximize Interactions with Them." In: *Modelling and Simulation for Autonomous Systems: Third Int. Workshop,* *MESAS 2016, Revised Selected Papers*. Ed. by Jan Hodicky. Springer International Publishing, pp. 48–63. ISBN: 978-3-319-47605-6.

Laugier, Christian, Stéphane Petti, Dizan Vasquez, Manuel Yguel, Thierry Fraichard, and Olivier Aycard (2007). "Steps towards safe navigation in open and dynamic environments." In: *Autonomous Navigation in Dynamic Environments*. Springer, pp. 45–82.

Liao, Lin, Dieter Fox, Jeffrey Hightower, Henry Kautz, and Dirk Schulz (2003). "Voronoi tracking: Location estimation using sparse and noisy sensor data." In: *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. Vol. 1, 723–728 vol.1.

Mehdi, Syed and Karsten Berns (2014). "Behavior-based search of human by an autonomous indoor mobile robot in simulation." In: *Universal Access in the Information Society* 13.1, pp. 45–58.

Moors, Mark, Timo Röhling, and Dirk Schulz (2005). "A probabilistic approach to coordinated multi-robot indoor surveillance." In: *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 3447–3452.

Morales, Yoichi, Satoru Satake, Rajibul Huq, Dylan Glas, Takayuki Kanda, and Norihiro Hagita (2012). "How do people walk side-by-side? Using a computational model of human behavior for a social robot." In: *ACM/IEEE Int. Conf. on Human-Robot Interaction*.

Nascimento, Tiago, António Moreira, and André Conceição (2013). "Multi-robot nonlinear model predictive formation control: Moving target and target absence." In: *Robotics and Autonomous Systems* 61.12, pp. 1502–1515.

Nishimura, Soh, Hiroshi Takemura, and Hiroshi Mizoguchi (2007). "Development of attachable modules for robotizing daily items – Person following shopping cart robot." In: *Proc. of the IEEE Int. Conf. on Robotics and Biomimetics (ROBIO)*.

Pradhan, Ninad, Timothy Burg, Stan Birchfield, and Ugur Hasirci (2013). "Indoor navigation for mobile robots using predictive fields." In: *Proc. of the American Control Conference*.

Prassler, Erwin, Dirk Bank, and Boris Kluge (2002). "Key technologies in robot assistants: Motion coordination between a human and a mobile robot." In: *Transactions on Control, Automation and Systems Engineering* 4.1.

Rabiner, Lawrence and Biing-Hwang Juang (1986). "An introduction to hidden Markov models." In: *IEEE ASSP Magazine* 3.1, pp. 4–16. ISSN: 0740-7467.

Schwenk, Markus, Tiago Vaquero, Goldie Nejat, and Kai Arras (2014). "Schedule-based Robotic Search for Multiple Residents in a Retirement Home Environment." In: *Proc. of the National Conf. on Artificial Intelligence (AAAI)*, pp. 2571–2577.

Silver, David, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. (2016). "Mastering the game of Go with deep neural networks and tree search." In: *Nature* 529.7587, pp. 484–489.

Stiffler, Nicholas, Andreas Kolling, and Jason O'Kane (2017). "Persistent pursuit-evasion: the case of preoccupied pursuer." In: *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pp. 5027–5034.

Sutton, Richard and Andrew Barto (1998). *Introduction to reinforcement learning*. MIT Press. ISBN: 0262193981.

Suzuki, Ichiro and Masafumi Yamashita (1992). "Searching for a Mobile Intruder in a Polygonal Region." In: *SIAM Journal on Computing* 21.5, pp. 863–888.

Tipaldi, Gian and Kai Arras (2011). "I want my coffee hot! Learning to find people under spatio-temporal constraints." In: *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pp. 1217–1222.

Vasquez, Dizan (2016). "Novel planning-based algorithms for human motion prediction." In: *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pp. 3317–3322.

Volkhardt, Michael and Horst-Michael Gross (2013). "Finding people in home environments with a mobile robot." In: *Proc. of the European Conf. on Mobile Robots (ECMR)*, pp. 282–287.

Ziebart, Brian, Nathan Ratliff, Garratt Gallagher, Christoph Mertz, Kevin Peterson, James Bagnell, Martial Hebert, Anind Dey, and Siddhartha Srinivasa (2009). "Planning-based prediction for pedestrians." In: *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 3931–3936.